

§ R E G N E C E N T R A L E N

Dob. præc. arithmetic  
in GIER ALGOL III  
Ole Møller

Abteilung für Num. Analyse

J. Zachariassen

Dezember 1965

JZ/GH

## Doppelte Präzision Arithmetik in GIER ALGOL

### Zusammenfassung:

In dem Folgenden wird eine Reihe von neuen Standardprozeduren im GIER ALGOL System beschrieben. Diese Prozeduren führen vorgeschriebene Berechnungen mit doppelter Präzision aus und beinhalten folgende Funktionen:

Addition, Subtraktion, Multiplikation, Division, Vorzeichenwechsel, Absolutwert, Quadratwurzel ausziehen, Potenzerrhöhung, Sinus, Kosinus, Arcus Tangens, Logarithmus naturalis, Exponentialfunktion, Eingabe und Ausgabe.

Die Standardprozeduren sind von einer Gruppe bestehend aus T. Asmussen, I. Möller, Ole Möller, P. V. Villumsen und J. Zachariassen codiert worden. Die Arbeit ist von J. Zachariassen zurechtgelegt und geleitet worden.

### 1. Doppelvariable

Jede Berechnung, die mit doppelter Präzision ausgeführt werden soll, wird mit dem Ruf von einer der neuen Standardprozeduren ausgeführt. (Eine Standardprozedur ist eine Prozedur, die in den ALGOL-Compiler eingebaut ist; sie soll also nicht in dem ALGOL-Programm, das sie benutzt, deklariert werden). Die Standardprozeduren, die in diesem Zusammenhang aktuell sind, werden in Abschnitt 3 in Einzelheiten beschrieben.

Variable doppelter Präzision - hier Doppelvariable genannt - nehmen 2 Zellen ein, die immer unmittelbar nach einander stehen. In der ersten Zelle stehen der Exponent und die ersten 30 Bit (einschliesslich Vorzeichen) des normalisierten Zahlenteils. In der zweiten Zelle stehen die letzten 39 Bit des Zahlenteils, da das erste Bit der zweiten Zelle (Nr. 0) immer gleich 0 ist. Hierdurch wird der Zahlenteil durch etwa 20 dezimale Ziffern dargestellt, und der Zahlenbereich wird dem der gewöhnlichen fliessenden Zahlen in GIER entsprechen.

In einem ALGOL-Programm, das doppelte Präzision verwendet, werden 2 Variable gefordert,

um einen Doppelvariablen zu enthalten. Ein Doppelvariabler kann also aus zwei einfachen Variablen oder zwei indizierten Variablen bestehen. In den Deklarationen sollen die 2 Variablen unmittelbar nach einander stehen, so dass der Variable, der den Exponententeil enthält, zuerst steht. Für indizierte Variable kann man z.B. eine zusätzliche Dimension haben. Auf diese Weise kann ein Doppelvariabler immer durch den ersten der zwei einfachen Variablen identifiziert werden, und der Wechsel von doppelter Präzision zu gewöhnlicher Arithmetik in ALGOL - und umgekehrt - wird besonders bequem.

Mit den Deklarationen

real      $x, x1, y, y1;$

array     $A [ 1 : n, 1 : m, 1 : 2 ], B [ 1 : p, 1 : 2 \times q ]$

kann man die  $n \times m + 2$  Doppelvariablen  $x, y$  und  $A [ i, j, 1 ], 1 \leq i \leq n, 1 \leq j \leq m$  sowie die  $p \times q$  Doppelvariablen  $B [ i, 2 \times j ], 1 \leq i \leq p, 1 \leq j \leq q$  anwenden.

## 2. Umformung vom Einfachvariablen zum Doppelvariablen und umgekehrt

Wenn man in einem ALGOL-Programm Berechnungen in gewöhnlicher Arithmetik ausführt, und möchte man an einer Stelle zu doppelter Präzision übergehen, kann es notwendig sein, "einen Doppelvariablen einem Einfachvariablen gleich zu setzen". Dies geschieht dadurch, dass man den zweiten Teil des Doppelvariablen gleich 0 und den ersten Teil gleich dem Einfachvariablen setzt.

Wenn man z.B. die Deklarationen

real      $x, y, a, a1, b, b1;$

hat, wo  $x$  und  $y$  als Einfachvariable behandelt worden sind, können die Werte von  $x$  und  $y$  mit Hilfe von folgenden Sätzen in die Doppelvariablen  $a$  und  $b$  "versetzt" werden:

$a := x; \quad a1 := 0;$

$b := y; \quad b1 := 0;$

Es ist aber unzweckmässig  $x$  (oder  $y$ ) als Doppelvariablen zu verwenden, da  $y$  (oder  $a$ ) dann als zweiter Teil des Doppelvariablen behandelt werden wird. Es wird vernünftig sein, Variable vom Typ real anzuwenden, weil es sonst unerwünschte Wirkungen haben kann (z.B. Abrundung bei integers).

Will man umgekehrt von doppelter Präzision zu einfacher Präzision übergehen und so einen Einfachvariablen einem Doppelvariablen gleich setzen, soll man bloss den Einfachvariablen dem ersten Teil des Doppelvariablen gleich setzen.

Wenn man die Deklarationen

real x, y, a, al, b, bl; (mit derselben Bedeutung wie vorher)

wie oben hat, können die Umformungen durch die Sätze

x: = a;

y: = b;

ausgeführt werden.

Beachten Sie, dass hierbei keine Abrundung sondern nur eine Abkürzung auf 30 Bit der Mantisse (der erste Teil des Doppelvariablen) geschieht.

Will man schliesslich 2 Doppelvariable gleich einander setzen, kann dies entweder mit Hilfe von der Standardprozedur assign (siehe unten, Abschnitt 3), oder, wenn die 2 Doppelvariablen (alle 4 Einfachvariablen) desselben Typ sind, durch die Sätze

a: = b; al: = bl;

geschehen, wenn es sich um die 2 Doppelvariablen a und b handelt.

Wenn die "rechte Seite" ein Prozedurenruf oder Expression ist, muss man aber immer die Standardprozedur assign anwenden.

### 3. Standardprozeduren

In diesem Abschnitt wird jede der neuen Standardprozeduren für sich beschrieben. Die Namen der Prozeduren, die Typenprozeduren (vom Typ real) sind, beinhalten den ersten Teil des Resultats (in Wirklichkeit die Zelle im Running System, die normalerweise Universal Value genannt wird), während der zweite Teil des Resultats in der unmittelbar nachfolgenden Zelle steht (Universal Value 1). Der Prozedurenname kann also als ein Doppelvariabler aufgefasst werden.

#### 3.1 add (a, b)

real procedure mit 2 Parametern; add fasst a und b als Doppelvariable auf und

errechnet die Summe von  $a$  und  $b$  ( $add := a + b$ ). Wenn der fließende Zahlenbereich überschritten wird, erhält man die Fehlerausschrift "spill".

### 3.2 sub (a, b)

real procedure mit 2 Parametern; sub fasst  $a$  und  $b$  als Doppelvariable auf und errechnet die Differenz zwischen  $a$  und  $b$  ( $sub := a - b$ ). Wenn der fließende Zahlenbereich überschritten wird, erhält man die Fehlerausschrift "spill".

### 3.3 mult (a, b)

real procedure mit 2 Parametern; mult fasst  $a$  und  $b$  als Doppelvariable auf und errechnet das Produkt von  $a$  und  $b$  ( $mult := a \times b$ ). Wenn der fließende Zahlenbereich überschritten wird, erhält man die Fehlerausschrift "spill".

### 3.4 div (a, b)

real procedure mit 2 Parametern; div fasst  $a$  und  $b$  als Doppelvariable auf und errechnet den Quotienten zwischen  $a$  und  $b$  ( $div := a/b$ ). Wenn der fließende Zahlenbereich überschritten wird, (z.B. durch Division mit 0), erhält man die Fehlerausschrift "spill".

### 3.5 minus (a)

real procedure mit 1 Parameter; minus fasst  $a$  als Doppelvariablen auf und wendet das Vorzeichen von  $a$  ( $minus := -a$ ). Wenn daraus folgt, dass der fließende Zahlenbereich überschritten wird, erhält man die Fehlerausschrift "spill". (Das Intervall des Zahlenteils einer fließenden Zahl ist je nach links abgeschlossen und nach rechts offen).

### 3.6 double abs (a)

real procedure mit 1 Parameter; double abs fasst  $a$  als Doppelvariablen auf und errechnet den numerischen Wert von  $a$ . Wenn der fließende Zahlenbereich durch einen Vorzeichenwechsel überschritten wird, erhält man die Fehlerausschrift "spill".

### 3.7 double sqrt (a)

real procedure mit 1 Parameter; double sqrt fasst  $a$  als Doppelvariablen auf und

errechnet die Quadratwurzel von  $a$ . Wenn  $a < 0$  ist, erhält man die Fehlerausschrift "sqrt",

### 3.8 power (a, n)

real procedure mit 2 Parametern. Der Wert des Parameters  $n$  muss mit einfacher Präzision und  $\geq 0$  sein; `power` rundet  $n$  auf integer ab, fasst  $a$  als Doppelvariablen auf und erhöht  $a$  zur Potenz  $n$  (`power` =  $a^n$ ). Wenn  $n < 0$  ist, erhält man die Fehlerausschrift "param", während man die Fehlerausschrift "spill" erhält, wenn der fließende Zahlenbereich überschritten wird.

### 3.9 double exp (a)

real procedure mit 1 Parameter; `double exp` fasst  $a$  als Doppelvariablen auf und errechnet die Exponentialfunktion zum Argument  $a$  (`double exp` =  $e^a$ ). Wenn hierdurch der fließende Zahlenbereich überschritten wird, erhält man die Fehlerausschrift "exp",

### 3.10 double ln (a)

real procedure mit 1 Parameter; `double ln` fasst  $a$  als Doppelvariablen auf und errechnet den Logarithmus naturalis von  $a$  (`double ln` =  $\log_e(a)$ ). Wenn  $a < 0$  ist, erhält man die Fehlerausschrift "ln",

### 3.11 double sin (a)

real procedure mit 1 Parameter; `double sin` fasst  $a$  als Doppelvariablen auf und errechnet Sinus von  $a$  (`double sin` =  $\sin(a)$ ).

### 3.12 double cos (a)

real procedure mit 1 Parameter; `double cos` fasst  $a$  als Doppelvariablen auf und errechnet Kosinus von  $a$  (`double cos` =  $\cos(a)$ ).

### 3.13 double arctan (a)

real procedure mit 1 Parameter; `double arctan` fasst  $a$  als Doppelvariablen auf und errechnet den Wert von Arcus Tangens zu  $a$ , der im Intervall  $-\frac{\pi}{2}$  bis  $\frac{\pi}{2}$  liegt (`double arctan` =  $\arctan(a)$ ).

### 3.14 assign (a, b)

real procedure mit 2 Parametern. Das aktuelle Parameter für a soll der Identifikator eines einfachen oder indizierten Variablen sein; assign fasst a und b als Doppelvariable auf und setzt den ersten Teil von a dem ersten Teil von b gleich, und den zweiten Teil von a dem zweiten Teil von b gleich; assign wird danach denselben Wert wie a haben (assign = a = b).

### 3.15 double input (a, b, c, ...)

procedure mit mehreren Parametern; double input funktioniert genau wie die Standard-prozedur input, die Parameter werden bloss als Doppelvariable aufgefasst, und man kann Zahlen mit bis zu 20 bedeutenden Ziffern einlesen. Beachten Sie, dass beim Einlesen von array eine zusätzliche Dimension, oder dergleichen, in den Deklarationen notwendig ist. (Siehe "A Manual of GIER ALGOL III".)

### 3.16 double inone

real procedure ohne Parameter. Die Prozedur funktioniert genau wie die Standard-prozedur inone, bloss wird der Prozedurenname "double inone" als Doppelvariablen aufgefasst und kann so in arithmetischen Ausdrücken mit doppelter Präzision angewandt werden; double inone kann also einlesen und bis zu 20 bedeutenden Ziffern enthalten.

### 3.17 double type in

real procedure ohne Parameter. Die Prozedur funktioniert wie double inone, bloss wird input von der Schreibmaschine genommen. Beachten Sie, dass die Zahlen, die die Prozeduren double input, double inone und double type in lesen können, den Konventionen gewöhnlicher ALGOL-Zahlen entsprechen müssen. Beachten Sie auch, dass die Fazilitäten char, setchar, input ditto, usw., genau auf dieselbe Weise von den obenstehenden input-Prozeduren wie von den Zahleneinleseprozeduren, die in "A Manual of GIER ALGOL III" beschrieben sind, behandelt werden.

### 3.18 double output (n, a, b, ...)

procedure mit mehreren Parametern. Das erste Parameter, n, wird als Einfachvariablen vom Typ integer aufgefasst und gibt an, mit wievielen Dezimalen die Doppelvariablen

a, b, ... gedruckt werden sollen. Dies ist die einzige Einteilungsmöglichkeit, und die Doppelvariablen a, b, ... werden mit 1 Ziffer vor dem Dezimalpunkt und n Dezimalen gedruckt; output wird immer auf dem Locher ausgegeben. Ausser obenerwähntem sind die Konventionen der aktuellen Parameter, check sum, usw., dieselben wie die der Standardprozeduren output. Es ist also möglich, einen Ruf von einer willkürlichen output-Prozedur (einschliesslich double output) als aktuellen Parameter anzuwenden.

### 3.19 double write (n, a, b, ...)

procedure mit mehreren Parametern. Die Prozedur fasst a, b, ... als Doppelvariable auf und druckt diese auf die Schreibmaschine mit n Dezimalen der angegebenen Reihenfolge nach. Ausser obenerwähntem funktioniert double write wie die Standardprozedur write (siehe Abschnitt 3.18 und "A Manual of GIER ALGOL III").

## 4. Aktuelle Parameter

Wenn die Anzahl aktueller Parameter in einem Ruf von einer der obenerwähnten Standardprozeduren nicht korrekt ist, erhält man die Fehlerausschrift

param

ausgenommen double input, double inone, double type in, double output und double write. Diese Fehlerausschrift wird man nicht während des Übersetzens des ALGOL-Programmes, das diese Prozeduren anwenden, sondern erst während der Ausführung des falschen Prozedurenruf - während des Programmablaufes - erhalten. (Siehe Abschnitt 3.8 und "A Manual of GIER ALGOL III", Seite 40).

Wo nichts Spezielles in Abschnitt 3 angegeben ist, kann man folgende als aktuelle Parameter in einem Ruf von einer der erwähnten Prozeduren anwenden:

- |    |   |  |
|----|---|--|
| a) | Doppelvariable  | (z.B.: assign (a, b))                            |
| b) | Zahlkonstante   | (z.B.: add (a, 1))                               |
| c) | Expressions   | (z.B.: add (2 x a, 4 x b))                       |
| d) | Ruf von einem willkürlichen Typ<br>Prozedur                                       | (z.B.: mult (sqrt (x), y))                       |
| e) | Ruf von einer der in Abschnitt 3<br>erwähnten Prozeduren, even-<br>tuell rekursiv | (z.B.: assign (x, add (sub (a, mult (b, 3)), y)) |



## 5. Relationen

Relationen vom Typ  $a < 0$ ,  $a \leq 0$ ,  $a = 0$ ,  $a \neq 0$ ,  $a \geq 0$  und  $a > 0$  werden auf gewöhnliche Weise geschrieben, während die Relationen vom Typ  $a < b$ ,  $a \leq b$ ,  $a = b$ ,  $a \neq b$ ,  $a \geq b$  und  $a > b$  mit Hilfe von obenstehenden Prozeduren umschrieben werden müssen, z.B.:

if sub (b, a) > 0 then .....

## 6. Compiler-Streifen

In "A Manual of GIER ALGOL III", Seite 33, ist beschrieben, wie es möglich ist, mit Hilfe von 5 separaten Streifen verschiedene binäre Versionen des Compilers herzustellen. Um auch eine binäre Version mit den in Abschnitt 3 erwähnten Prozeduren herstellen zu können, ist es notwendig, neue Fassungen der Streifen A, B, C und D anzuwenden.

### 6.1 Compiler doppelter Präzision

Die Herstellung einer binären Version eines solchen Compilers erfordert folgendes Verfahren:

- a) Einlesen von Streifen A mit Hilfe von SLIP (mit I starten).  
Nach ein paar Sekunden stoppt die Maschine, und man muss nun die Redefinition

e68 = 1

vornehmen, wonach das Einlesen mit I fortgesetzt wird. Wenn man wünscht, den Compiler an einer anderen Stelle als von Kanal 39 anzubringen, kann man an dieser Stelle die Namen e96, c60, c70, usw., redefinieren.

- b) Wenn Streifen A eingelesen worden ist, wird Streifen B (oder Streifen C für eine transiente Version) eingelesen.  
c) Nach dem Einlesen des letzten Streifens schreibt die Maschine algol.  
d) Nun wird das ALGOL-Programm eingelesen:

begin gierproc ( << binout >, 5) end;

Dieses Programm fordert Streifen E als input.

In der Anlage gibt es ALGOL-Programme zur Veranschaulichung der Verwendung der doppelten Präzision.

```
begin  
procedure Test(a); string a;  
begin  
    writecr; writetext(a); outsp(50); outtext(<<  
Test >,a)  
end;
```

comment Test programs 10.127 - 10.136 are inserted here;

end Test 10.127 - 10.136;

```
data double ln:      0.43429 44819 03251 82765, 2.71828 18284 59045 23536  
data double sin:     3.14159 26535 89793 23835, 3.14159 26535 89793 23855  
data double cos:     3.14159 26535 89793 23835, 3.14159 26535 89793 23855  
data double arctan:  3.14159 26535 89793 23835, 3.14159 26535 89793 23855
```

```

Test({<10.127});
begin integer i,j,j1;
  real ul, arg, a1, max, m1, maxarg, ma1, exlog, e1, r1, r11, r2, r21, h, h1, rm, ru;
  outtext({< double ln, double exp, exlog}, outcr, outcr, outcr,
    {<arg1 ln(arg1)}, outsp(23), {<rel.error}, outcr);
  for i:=100 step 50 until 300 do
    begin
      max:=m1:=0; j1:=0;
      for j:=i-50 step 1 until 1 do
        begin
          assign(arg, div(j, 100));
          assign(exlog, doubleexp(doubleln(arg)));
          assign(h, sub(arg, exlog));
          if abs(h)>abs(max) then
            begin assign(max, h); assign(maxarg, arg) end
          end;
          outcr; output({d.dd}, maxarg, outsp(3));
          doubleoutput(20, doubleln(maxarg), outsp(3));
          output({-d.d10+dd}, div(max, maxarg))
        end;
      outcr; outcr;
      outtext({<arg2}, outsp(29), {<ln(arg2)}, outsp(24),
        {<abs.error2 rel.error2}, outcr);
      assign(r1, div(1, mult(power(10, 64), power(10, 76))));
      for j:=j while r1<10125 do
        begin
          max:=m1:=0; ul:=21019xr1; assign(r2, r1);
          for j:=j while r2<ul do
            begin
              assign(exlog, doubleexp(doubleln(r2)));
              assign(h, sub(r2, exlog));
              if sub(double abs(h), doubleabs(max))>0 then
                begin assign(max, h); assign(maxarg, r2) end;
              assign(r2, mult(10, r2))
            end;
          outcr;
          doubleoutput(20, maxarg, outsp(5), doubleln(maxarg), outsp(5));
          output({-d.d10+dd}, div(max, maxarg), outsp(4),
            div(div(max, maxarg), doubleln(maxarg)));
          assign(r1, if 0.9<r1/r1<1.1^-., (r1=1/r11=0) then 1 else
            mult(r1, power(10, 20)))
        end;
      outchar(42)
    end;
  end;

```

Test 10.127      double ln, double exp, exlog

arg1	ln(arg1)		rel.error
.95	-5.12932943875505333542 <sub>10</sub>	2	-7.1 <sub>10</sub> -20
1.47	3.85262400790644933614 <sub>10</sub>	1	-4.6 <sub>10</sub> -20
1.93	6.57520002916794183895 <sub>10</sub>	1	-7.4 <sub>10</sub> -20
2.38	8.67100487683383326759 <sub>10</sub>	1	-2.8 <sub>10</sub> -20
2.83	1.04027671165514625664		-4.3 <sub>10</sub> -20

arg2		ln(arg2)		abs.error2	rel.error2
9.99999999999999980 <sub>10</sub>	-123	-2.80915381345273573451 <sub>10</sub>	2	2.6 <sub>10</sub> -18	-9.1 <sub>10</sub> -21
9.99999999999999980 <sub>10</sub>	-102	-2.32561094392398614086 <sub>10</sub>	2	7.5 <sub>10</sub> -19	-3.2 <sub>10</sub> -21
9.99999999999999960 <sub>10</sub>	-82	-1.86509392532517700406 <sub>10</sub>	2	3.9 <sub>10</sub> -19	-2.1 <sub>10</sub> -21
9.99999999999999964 <sub>10</sub>	-62	-1.40457690672636786725 <sub>10</sub>	2	1.3 <sub>10</sub> -18	-8.9 <sub>10</sub> -21
9.99999999999999959 <sub>10</sub>	-42	-9.44059888127558730448 <sub>10</sub>	1	-3.0 <sub>10</sub> -19	3.2 <sub>10</sub> -21
9.99999999999999958 <sub>10</sub>	-22	-4.83542869528749593645 <sub>10</sub>	1	-4.0 <sub>10</sub> -20	8.3 <sub>10</sub> -22
9.99999999999999958 <sub>10</sub>	-2	-2.30258509299104568402		-4.7 <sub>10</sub> -20	2.0 <sub>10</sub> -20
9.99999999999999944 <sub>10</sub>	18	4.37491167668868679962 <sub>10</sub>	1	1.6 <sub>10</sub> -19	3.7 <sub>10</sub> -21
1.0000000000000000000 <sub>10</sub>	19	4.37491167668868679962 <sub>10</sub>	1	2.2 <sub>10</sub> -19	5.0 <sub>10</sub> -21
9.99999999999999979 <sub>10</sub>	38	8.98008186267677816763 <sub>10</sub>	1	1.5 <sub>10</sub> -19	1.7 <sub>10</sub> -21
9.99999999999999962 <sub>10</sub>	58	1.35852520486648695357 <sub>10</sub>	2	3.9 <sub>10</sub> -19	2.9 <sub>10</sub> -21
9.99999999999999963 <sub>10</sub>	78	1.81904222346529609036 <sub>10</sub>	2	5.0 <sub>10</sub> -20	2.8 <sub>10</sub> -22
9.99999999999999959 <sub>10</sub>	98	2.27955924206410522717 <sub>10</sub>	2	9.0 <sub>10</sub> -19	4.0 <sub>10</sub> -21
9.99999999999999950 <sub>10</sub>	118	2.74007626066291436396 <sub>10</sub>	2	3.0 <sub>10</sub> -18	1.1 <sub>10</sub> -20
9.99999999999999968 <sub>10</sub>	138	3.20059327926172350078 <sub>10</sub>	2	2.3 <sub>10</sub> -19	7.1 <sub>10</sub> -22

```

Test(<<10.128>);
begin
integer q,q1,y,y1,upy,lowy,maxq,maxupy,maxlowy;
real max,m1,up,u1,low,l1,log,l2,dif,d1,M,M1,e,e1,h,h1;
procedure print(u,u1); value u,u1; real u,u1;
begin
real w,w1;
assign(w,doubleln(u)); outcr;
output(<dd.dd>,u,outsp(2));
doubleoutput(20,w,outsp(2),mult(w,M))
end;

doubleinput(M,e);
outtext(<< double ln>,outcr,outcr,outcr);
max:=-1; m1:=0; q1:=y1:=0;
for q:=11 step 1 until 19,20 step 10 until 90 do
begin
assign(up,sub(doubleln(div(q,10)),1)); assign(low,add(up,2));
for y:=1 step 1 until 100 do
begin
assign(log,sub(doubleln(div(qxy,100)),doubleln(div(y,10))));
if sub(log,up)>0 then begin upy:=y; assign(up,log) end;
if sub(log,low)<0 then begin lowy:=y; assign(low,log) end
end;
assign(dif,sub(up,low));
if sub(dif,max)>0 then
begin maxq:=q; maxupy:=upy; maxlowy:=lowy; assign(max,dif) end
end;
outcr;
output(<nd.d>,maxq/10); outsp(4); output(<d.dn+dd>,max); outcr; outcr;
outtext(<< x ln(x) logn(x)>);
for y:=maxlowy,maxupy do
begin outcr; assign(h,div(maxqxy,100)); print(h,h1);
assign(h,div(y,10)); print(h,h1)
end;
outcr; print(1,0); outcr; print(e,e1); outcr; print(10,0);
outcr; outchar(42)
end;

comment data

M 0.43429 44819 03251 82765
e 2.71828 18284 59045 23536;

```

Test 10.128 double ln

9.0	1.4 <sub>10</sub> -19	
x	ln(x)	log <sub>10</sub> (x)
66.60	4.19870457754634345253	1.82347422917030106658
7.40	2.00148000021012406982	8.69231719730976192043 <sub>10</sub> - 1
.90	-1.05360515657826301166 <sub>10</sub> - 1	-4.57574905606751253832 <sub>10</sub> - 2
.10	-2.30258509299404568399	-9.99999999999999999986 <sub>10</sub> - 1
1.00	-3.38813178901720135627 <sub>10</sub> - 21	-1.47144693993116319304 <sub>10</sub> - 21
2.72	1.00000000000000000000.2	4 34294481903251827659 <sub>10</sub> - 1
10.00	2 30258509299404568402	9.99999999999999999996 <sub>10</sub> - 1

```

Test(<<10.129>>);
begin integer k;
real arg,arg1,max,max1,maxarg,maxarg1,konst,konst1,hj,hj1,
basearg,basearg1,relative error,relative error 1;
outtext(<< double sqrt>>,outcr,outcr,outcr,<<arg>>,outsp(28),
<<sqrt(arg)>>,outsp(22),<<rel.error>>,outcr);
assign(konst,div(11,10));
for k:=-100 step 50 until 150 do
begin
assign(basearg,if k<0 then div(1,power(10,-k)) else
if k<150 then power(10,k) else
mult(power(10,64),power(10,86)));
max :=-1; max1:=0;
assign(arg,div(basearg,power(10,50)));
for k:=k while sub(arg,basearg)<0 do
begin
assign(hj,doublesqrt(arg));
assign(relative error,div(doubleabs(sub(mult(hj,hj),arg)),arg))
if sub(relative error,max)>0 then
begin
assign(max,relative error);
assign(maxarg,arg)
end;
assign(arg,mult(arg,konst))
end;
doubleoutput(20,outcr,maxarg,outsp(3),doublesqrt(maxarg),outsp(3));
output(<<d.dp-dd>>,max);
end;
outchar(42)
end;

```

Test 10.129      double sqrt

arg	sqrt(arg)	rel.error
2.60081036420023296397 <sub>10</sub> -125	5.09981407916036970887 <sub>10</sub> - 63	1.8 <sub>10</sub> -20
1.66164735807218281035 <sub>10</sub> - 58	1.28904901306047428753 <sub>10</sub> - 29	1.9 <sub>10</sub> -20
1.92802467550106259887 <sub>10</sub> - 43	4.39092777838700348492 <sub>10</sub> - 22	1.9 <sub>10</sub> -20
6.96536955806386640892 <sub>10</sub> 4	2.63919865831730567906 <sub>10</sub> 2	1.9 <sub>10</sub> -20
2.05398207033957515888 <sub>10</sub> 78	1.43317203096473214660 <sub>10</sub> 39	1.8 <sub>10</sub> -20
8.30956521921224821199 <sub>10</sub> 149	9.11568166360160265326 <sub>10</sub> 74	2.0 <sub>10</sub> -20



```

Test(<<10.130>>);
begin real x,x1,h,h1;
  outtext(<< double exp>>,outcr,outcr,outcr,<<arg>>,
    outsp(27),<<exp(arg)>>,outcr,outcr);
  doubleoutput(20,0,outsp(2),doubleexp(0));
  x:=-1; x1:=0;
  for x:=2*x while x>-1050 do
    begin
      assign(h,doubleexp(x));
      if h<0 then
        begin
          outcr; doubleoutput(20,x,outsp(2),h,outcr);
          assign(h,doubleexp(mult(3,x)/4));
          doubleoutput(20,mult(3,x)/4,outsp(2),h);
          x:=-1; x1:=0; goto B
        end
      end;
    end;
  B:
  for x:=x/2 while abs(x)>10-20 do
    begin
      assign(h,doubleexp(x));
      if h=1^h1=0 then
        begin
          outcr; doubleoutput(20,x,outsp(2),h,outcr,
            mult(3,x)/2,outsp(2),
            doubleexp(mult(3,x)/2));
          if x<0
            then begin x:=1; goto B end
            else goto A
          end
        end;
      end;
    end;
  A:
  for x:=1,xx2 while x<512 do
    doubleoutput(20,outcr,x,outsp(2),doubleexp(x));
    outchar(42)
  end;
end;

```

Test 10.130 double exp

arg		exp(arg)	
0.00000000000000000000		9.999999999999999999 <sub>10</sub>	1
-5.120000000000000000 <sub>10</sub>	2	0.00000000000000000000	
-3.839999999999999999 <sub>10</sub>	2	0.00000000000000000000	
1.00000000000000000000		2.71828182845904523536	
2.00000000000000000000		7.38905609893065022723	
4.00000000000000000000		54.5981500331442390781 <sub>10</sub>	1
8.00000000000000000000		2.98095798704172827474 <sub>10</sub>	3
1.599999999999999999 <sub>10</sub>	1	8.88611052050787263681 <sub>10</sub>	6
3.200000000000000000 <sub>10</sub>	1	7.89629601826806951618 <sub>10</sub>	13
6.400000000000000000 <sub>10</sub>	1	6.23514908081161688306 <sub>10</sub>	27
1.279999999999999999 <sub>10</sub>	2	3.88770840599459509242 <sub>10</sub>	55
2.560000000000000000 <sub>10</sub>	2	1.51142766500410354268 <sub>10</sub>	111

```

Test(<<10.131>);
begin
integer n,n1,i,j,j1;
real max,m1,arg,a1,plog,p11,ppow,pp1,h,h1,maxarg,mal;
outtext(<< double exp,double ln>,outcr,outcr,outcr,<<n>,outsp(4),
<<arg>,outsp(4),<<abs error>,outsp(3),<<rel error>,outcr);

n1:=0;
for n:=2,3,5,10 do
begin
outcr; output(<nd>,n,outsp(3));
for i:=100 step 50 until 300 do
begin
max:=m1:=0; j1:=0;
for j:=i-50 step 1 until i do
begin
assign(arg,div(j,100));
assign(plog,doubleexp(mult(n,doubleln(arg))));
assign(ppow,power(arg,n));
assign(h,sub(plog,ppow));
if abs(h)>abs(max) then
begin assign(max,h); assign(maxarg,arg) end
end;
output(<d.dd>,maxarg); outsp(3);
assign(h,div(max,power(maxarg,n)));
output(<-d.dd10+dd>,max,outsp(3),h,outcr,outsp(5))
end;
outcr
end;

doubleoutput(20,outcr,doubleln(doubleexp(1)),outcr,
doubleln(doubleln(doubleexp(doubleexp(div(1,100))))));
outchar(42)
end;

```

Test 10.131 double exp, double ln

n	arg	abs error	rel error
2	.95	$1.32_{10^{-19}}$	$1.46_{10^{-19}}$
	1.47	$2.03_{10^{-19}}$	$9.41_{10^{-20}}$
	1.93	$5.83_{10^{-19}}$	$1.56_{10^{-19}}$
	2.39	$3.52_{10^{-19}}$	$6.17_{10^{-20}}$
	2.94	$7.59_{10^{-19}}$	$8.78_{10^{-20}}$
3	.97	$1.93_{10^{-19}}$	$2.12_{10^{-19}}$
	1.47	$4.20_{10^{-19}}$	$1.32_{10^{-19}}$
	1.99	$1.73_{10^{-18}}$	$2.20_{10^{-19}}$
	2.47	$1.14_{10^{-18}}$	$7.55_{10^{-20}}$
	3.00	$3.14_{10^{-18}}$	$1.16_{10^{-19}}$
5	.99	$3.12_{10^{-19}}$	$3.28_{10^{-19}}$
	1.47	$1.49_{10^{-18}}$	$2.17_{10^{-19}}$
	1.99	$1.12_{10^{-17}}$	$3.58_{10^{-19}}$
	2.47	$1.26_{10^{-17}}$	$1.37_{10^{-19}}$
	2.99	$4.51_{10^{-17}}$	$1.89_{10^{-19}}$
10	.99	$5.93_{10^{-19}}$	$6.56_{10^{-19}}$
	1.50	$2.10_{10^{-17}}$	$3.65_{10^{-19}}$
	1.99	$6.97_{10^{-16}}$	$7.16_{10^{-19}}$
	2.47	$2.28_{10^{-15}}$	$2.69_{10^{-19}}$
	3.00	$2.15_{10^{-14}}$	$3.65_{10^{-19}}$

1.00000000000000000003

1.0000000000000000000353<sub>10<sup>-2</sup></sub>

```

Test(<<10.132>>);
begin integer i,j,jl;
  real arg,a1,max,m1,maxarg,mal,logex,el,h,h1;
  outtext(<< double ln, double exp, logex>>,outer,outer,outer,
    <<arg ln(exp(arg))>>,outsp(18),<<rel.error>>,outer);
  for i:=100 step 50 until 300 do
    begin
      max:=m1:=0; jl:=0;
      for j:=1-50 step 1 until 1 do
        begin
          assign(arg,div(j,100));
          assign(logex,doubleln(doubleexp(arg)));
          assign(h,sub(arg,logex));
          if abs(h)>abs(max) then
            begin assign(max,h); assign(maxarg,arg) end
          end;
          outer; output(<d.dd>,maxarg,outsp(3));
          doubleoutput(20,doubleln(doubleexp(maxarg))); outsp(3);
          output(<-d.d10+dd>, max/maxarg)
        end;
      outchar(42)
    end;
end;

```

Test 10.132 double ln, double exp, logex

arg	ln(exp(arg))	rel.error
.67	6.70000000000000000000000069 <sub>10</sub> - 1	-1.1 <sub>10</sub> -19
1.34	1.34000000000000000000000007	-5.6 <sub>10</sub> -20
1.89	1.89000000000000000000000006	-3.6 <sub>10</sub> -20
2.03	2.03000000000000000000000006	-4.0 <sub>10</sub> -20
2.77	2.77000000000000000000000007	-2.9 <sub>10</sub> -20

```

Test(<<10.133>);
begin integer i, i1, k, k1;
real s, s1, x, x1, led, led1, de, del;
  outtext(<< double exp>, outcr, outcr, outcr, <<x>, outsp(6),
    <<exp(x) - Taylor>, outsp(14), <<exp(x)>, outsp(27),
    <<abs.error rel.error>, outcr);
  i1:=k1:=0;
  for i:=-50,-20,-15,-10 step 1 until 10, 15,20,50 do
    begin
      assign(s,1);
      assign(led,s);
      assign(x,div(i,10));
      k:=2;
L:    k:=k+1;
      assign(led,div(mult(x,led),k));
      assign(s,add(s,led));
      if abs(led)>10-20 then goto L;
      assign(s,add(1,add(x,mult(mult(x,x),div(s,2)))));
      output(<-d.d>, outcr, x, outsp(2));
      assign(de,doubleexp(x));
      doubleoutput(20,s,outsp(2),de,outsp(4));
      output(<-d.d0-dd>, sub(s,de), outsp(4), div(sub(s,de),s));
    end;
  outchar(42);
end;

```

Test 10.133 double exp

x	exp(x) - Taylor	exp(x)		abs.error	rel.error
-5.0	6.73794699908546645874 <sub>10</sub> -	6.73794699908546709674 <sub>10</sub> -	3	-6.4 <sub>10</sub> -19	-9.5 <sub>10</sub> -17
-2.0	1.35335283236612691787 <sub>10</sub> -	1.35335283236612691893 <sub>10</sub> -	1	-1.1 <sub>10</sub> -19	-7.8 <sub>10</sub> -19
-1.5	2.23130160148429828907 <sub>10</sub> -	2.23130160148429828932 <sub>10</sub> -	1	-2.5 <sub>10</sub> -20	-1.1 <sub>10</sub> -19
-1.0	3.67879441171442321576 <sub>10</sub> -	3.67879441171442321594 <sub>10</sub> -	1	-1.9 <sub>10</sub> -20	-5.1 <sub>10</sub> -20
-.9	4.06369659740599111859 <sub>10</sub> -	4.06369659740599111881 <sub>10</sub> -	1	-2.2 <sub>10</sub> -20	-5.4 <sub>10</sub> -20
-.8	4.49328964117221591414 <sub>10</sub> -	4.49328964117221591427 <sub>10</sub> -	1	-1.4 <sub>10</sub> -20	-3.0 <sub>10</sub> -20
-.7	4.96585303791409514692 <sub>10</sub> -	4.96585303791409514703 <sub>10</sub> -	1	-1.2 <sub>10</sub> -20	-2.4 <sub>10</sub> -20
-.6	5.48811636094026432618 <sub>10</sub> -	5.48811636094026432624 <sub>10</sub> -	1	-6.8 <sub>10</sub> -21	-1.2 <sub>10</sub> -20
-.5	6.06530659712633423599 <sub>10</sub> -	6.06530659712633423602 <sub>10</sub> -	1	-3.4 <sub>10</sub> -21	-5.6 <sub>10</sub> -21
-.4	6.70320046035639300735 <sub>10</sub> -	6.70320046035639300741 <sub>10</sub> -	1	-6.8 <sub>10</sub> -21	-1.0 <sub>10</sub> -20
-.3	7.40818220681717866061 <sub>10</sub> -	7.40818220681717866065 <sub>10</sub> -	1	-3.4 <sub>10</sub> -21	-4.6 <sub>10</sub> -21
-.2	8.18730753077981858662 <sub>10</sub> -	8.18730753077981853669 <sub>10</sub> -	1	-6.8 <sub>10</sub> -21	-8.3 <sub>10</sub> -21
-.1	9.04837418035959573160 <sub>10</sub> -	9.048374180355959573164 <sub>10</sub> -	1	-3.4 <sub>10</sub> -21	-3.7 <sub>10</sub> -21
.0	1.00000000000000000000	9.9999999999999999999996 <sub>10</sub> -	1	.0	.0
.1	1.10517091807564762480	1.10517091877564762481		-6.8 <sub>10</sub> -21	-6.1 <sub>10</sub> -21
.2	1.22140275816016983391	1.22140275816016983391		-6.8 <sub>10</sub> -21	-5.5 <sub>10</sub> -21
.3	1.34985880757600310397	1.34985880757600310397		.0	.0
.4	1.49182469764127031781	1.49182465734127031781		.0	.0
.5	1.64872127070012814683	1.64872127070012814684		-6.8 <sub>10</sub> -21	-4.1 <sub>10</sub> -21
.6	1.82211880039050897485	1.82211880039050897486		-6.8 <sub>10</sub> -21	-3.7 <sub>10</sub> -21
.7	2.01375270747047652159	2.01375270747047652162		-2.7 <sub>10</sub> -20	-1.3 <sub>10</sub> -20
.8	2.22554092849246760453	2.22554092849246760455		-1.4 <sub>10</sub> -20	-6.1 <sub>10</sub> -21
.9	2.45960311115694966376	2.45960311115694966378		-2.7 <sub>10</sub> -20	-1.1 <sub>10</sub> -20
1.0	2.71828182845904523533	2.71828182845904523536		-2.7 <sub>10</sub> -20	-1.0 <sub>10</sub> -20
1.5	4.48168907033806482249	4.48168907033806482258		-8.1 <sub>10</sub> -20	-1.8 <sub>10</sub> -20
2.0	7.38905609893065022691	7.38905609893065022723		-3.3 <sub>10</sub> -19	-4.4 <sub>10</sub> -20
5.0	1.48413159102576603408 <sub>10</sub>	1.48413159102576603417 <sub>10</sub>	2	-9.5 <sub>10</sub> -18	-6.4 <sub>10</sub> -20



```

Test({<10.134});
begin integer 1,i1,k,k1,m,j;
real pi,pi1,s,s1,x,x1,xh,xh1,xsq,xsq1,led,led1,max,max1,
maxh,maxh1,maxi,maxs,maxs1,maxx,maxx1;
  outtext({< double sin},outcr,outcr,outcr,{<pi},
    outsp(26),{<degrees sin(x) - Taylor},
    outsp(15),{<sin(x)},outsp(23),{<abs.error rel.error},outcr);
i1:=k1:=0;
for j:=1,2 do
  begin
    assign(pi,doubleinone);
    doubleoutput(20,outcr,outcr,pi);
    assign(xh,div(pi,1800));
    for m:=400 step 400 until 4000 do
      begin
        max:=max1:=0;
        for i:=m-400 step 5 until m do
          begin
            assign(x,mult(xh,i));
            assign(xsq,mult(x,x));
            assign(s,assign(led,div(xsq,20)));
            k:=4;
L:          k:=k+2;
            assign(led,div(div(mult(minus(xsq),led),k),k+1));
            assign(s,add(s,led));
            if abs(led)>mult(10-20,doubleabs(s)) then goto L;
            assign(s,add(x,div(mult(sub(s,1),mult(xsq,x)),6)));
            assign(maxh,doubleabs(sub(s,doublesin(x))));
            if sub(maxh,max)>0 then
              begin
                assign(max,maxh);
                maxi:=1;
                assign(maxs,s);
                assign(maxx,x)
              end
            end;
            output({<ndd.d},maxi/10,outsp(4));
            assign(x,maxx);
            assign(s,doublesin(x));
            doubleoutput(20,maxs,outsp(2),s,outsp(3));
            output({<-d.d40-dd},max,outsp(3),
              div(max,add(doubleabs(s),doubleabs(mult(x,doublecos(x))))));
            outcr; outsp(28)
          end;
        end;
      outchar(42)
    end;
  end;
comment data
3.14159265358979323835, 3.14159265358979323855;

```

Test 10.134 double sin

pi	degrees	sin(x) - Taylor	sin(x)	abs.error	rel.error
3.14159265358979323833	29.0	4.84809620246337029052 <sub>10</sub> -	4.84809620246337029055 <sub>10</sub> -	3.4 <sub>10</sub> -21	3.7 <sub>10</sub> -21
	78.0	9.78147600733805637919 <sub>10</sub> -	9.78147600733805637919 <sub>10</sub> -	1.0 <sub>10</sub> -20	8.1 <sub>10</sub> -21
	119.0	8.74619707139395800309 <sub>10</sub> -	8.74619707139395800326 <sub>10</sub> -	1.7 <sub>10</sub> -20	9.0 <sub>10</sub> -21
	159.0	3.58367949545300273531 <sub>10</sub> -	3.58367949545300273604 <sub>10</sub> -	7.3 <sub>10</sub> -20	2.5 <sub>10</sub> -20
	193.0	-2.24951054343864990359 <sub>10</sub> -	-2.24951054343864997899 <sub>10</sub> -	1.6 <sub>10</sub> -19	4.6 <sub>10</sub> -20
	235.0	-8.19152044288091790048 <sub>10</sub> -	-8.19152044288991789588 <sub>10</sub> -	4.6 <sub>10</sub> -19	1.5 <sub>10</sub> -19
	279.5	-9.86285601537231408799 <sub>10</sub> -	-9.86285601537231407870 <sub>10</sub> -	9.3 <sub>10</sub> -19	5.2 <sub>10</sub> -19
	316.5	-6.88354575693753985841 <sub>10</sub> -	-6.88354575693753984567 <sub>10</sub> -	1.3 <sub>10</sub> -18	2.7 <sub>10</sub> -19
	352.0	-1.39173100960065446548 <sub>10</sub> -	-1.3917310096006544388 <sub>10</sub> -	2.2 <sub>10</sub> -18	3.5 <sub>10</sub> -19
	399.5	6.36070220277103942873 <sub>10</sub> -	6.36078220277763946566 <sub>10</sub> -	3.7 <sub>10</sub> -18	6.1 <sub>10</sub> -19
3.14159265358979323853	29.5	4.92423560103467119803 <sub>10</sub> -	4.92423560103467119806 <sub>10</sub> -	3.4 <sub>10</sub> -21	3.6 <sub>10</sub> -21
	78.5	9.79924704620829611879 <sub>10</sub> -	9.79924704620829611889 <sub>10</sub> -	1.0 <sub>10</sub> -20	8.1 <sub>10</sub> -21
	119.0	8.74619707139395800242 <sub>10</sub> -	8.74619707139395800259 <sub>10</sub> -	1.7 <sub>10</sub> -20	9.0 <sub>10</sub> -21
	155.0	4.22618261740699436082 <sub>10</sub> -	4.22618261740699436146 <sub>10</sub> -	6.4 <sub>10</sub> -20	2.2 <sub>10</sub> -20
	186.0	-1.04528463267653471606 <sub>10</sub> -	-1.04528463267653471464 <sub>10</sub> -	1.4 <sub>10</sub> -19	4.3 <sub>10</sub> -20
	235.0	-8.19152044288991790184 <sub>10</sub> -	-8.19152044288991789726 <sub>10</sub> -	4.6 <sub>10</sub> -19	1.4 <sub>10</sub> -19
	279.5	-9.86285601537231408609 <sub>10</sub> -	-9.86285601537231407816 <sub>10</sub> -	7.9 <sub>10</sub> -19	4.4 <sub>10</sub> -19
	315.5	-7.00909264299850901408 <sub>10</sub> -	-7.00909264299850900100 <sub>10</sub> -	1.3 <sub>10</sub> -18	2.8 <sub>10</sub> -19
	355.0	-8.71557427476581757903 <sub>10</sub> -	-8.71557427476581734474 <sub>10</sub> -	2.3 <sub>10</sub> -18	3.7 <sub>10</sub> -19
	396.5	5.94822786751341286816 <sub>10</sub> -	5.94822786751341290671 <sub>10</sub> -	3.8 <sub>10</sub> -18	6.2 <sub>10</sub> -19

```

Test(<<10 135>);
begin integer i,i1,k,k1,m,j;
real pi,p1,s,s1,x,x1,xh,xh1,xsq,xsq1,led,led1,max,max1,
maxh,maxh1,maxi,maxs,maxs1,maxx,maxx1;
  outtext(<< double cos>,outcr,outcr,outcr,<<pi>,
    outsp(26),<<degrees cos(x) - Taylor>,
    outsp(15),<<cos(x)>,outsp(24),<<abs.error rel.error>,outcr);
i1:=k1:=0;
for j:=1,2 do
  begin
    assign(pi,doubleinone);
    doubleoutput(20,outcr,outcr,pi);
    assign(xh,div(pi,1800));
    for m:=400 step 400 until 4000 do
      begin
        max:=max1:=0;
        for i:=m-400 step 5 until m do
          begin
            assign(x,mult(xh,i));
            assign(xsq,mult(x,x));
            assign(s,assign(led,div(xsq,12)));
            k:=3;
            k:=k+2;
            assign(led,div(div(mult(minus(xsq),led),k),k+1));
            assign(s,add(s,led));
            if abs(led)>mult(10-20,doubleabs(s)) then goto L;
            assign(s,add(1,div(mult(sub(s,1),xsq),2)));
            assign(maxh,doubleabs(sub(s,doublecos(x))));
            if sub(maxh,max)>0 then
              begin
                assign(max,maxh);
                maxi:=i;
                assign(maxs,s);
                assign(maxx,x)
              end
            end;
            output(<<ddd.d>,maxi/10,outsp(4));
            assign(x,maxx);
            assign(s,doublecos(x));
            doubleoutput(20,maxs,outsp(2),s,outsp(3));
            output(<<-d.d,0-dd>,max,outsp(3),
              div(max,add(doubleabs(s),doubleabs(mult(x,doublesin(x))))));
            outcr; outsp(28)
          end;
        end;
      end;
    outchar(42)
  end;
comment data:
3.14159265358979323835, 3.14159265358979323855;

```

Test 10.135 double cos

pi	degrees	cos(x) - Taylor	cos(x)	abs.error	rel.error
3.14159265358979323835	13.5	9.72369920397676601835 <sup>-10</sup>	1	1.0 <sup>-20</sup>	9.9 <sup>-21</sup>
	68.0	3.74606593415912035451 <sup>-10</sup>	1	1.5 <sup>-20</sup>	1.0 <sup>-20</sup>
	108.0	-3.09016224274947424056 <sup>-10</sup>	1	3.4 <sup>-20</sup>	1.6 <sup>-20</sup>
	159.5	-9.36672189248397616312 <sup>-10</sup>	1	1.2 <sup>-19</sup>	6.0 <sup>-20</sup>
	194.0	-9.70295726275996472503 <sup>-10</sup>	1	2.2 <sup>-19</sup>	1.2 <sup>-19</sup>
	235.0	-5.7357613635106096611 <sup>-10</sup>	1	3.6 <sup>-19</sup>	9.1 <sup>-20</sup>
	279.5	1.65047605860677647581 <sup>-10</sup>	1	5.7 <sup>-19</sup>	1.1 <sup>-19</sup>
	315.0	7.07106781185547523271 <sup>-10</sup>	1	9.6 <sup>-19</sup>	2.1 <sup>-19</sup>
	360.0	9.9999999999999999997858 <sup>-10</sup>	1	2.1 <sup>-18</sup>	2.1 <sup>-18</sup>
	395.5	8.14115518356319215595 <sup>-10</sup>	1	3.0 <sup>-18</sup>	6.3 <sup>-19</sup>
3.14159265358979323853	13.0	9.7437006478525228538 <sup>-10</sup>	1	1.0 <sup>-20</sup>	9.9 <sup>-21</sup>
	68.0	3.74606593415912035388 <sup>-10</sup>	1	1.4 <sup>-20</sup>	9.2 <sup>-21</sup>
	106.0	-2.7563735581699185715 <sup>-10</sup>	1	3.2 <sup>-20</sup>	1.6 <sup>-20</sup>
	160.0	-9.39692620785908384181 <sup>-10</sup>	1	1.2 <sup>-19</sup>	6.3 <sup>-20</sup>
	200.0	-9.39692620785908384238 <sup>-10</sup>	1	2.1 <sup>-19</sup>	9.8 <sup>-20</sup>
	238.0	-5.29919264233214954356 <sup>-10</sup>	1	3.7 <sup>-19</sup>	9.0 <sup>-20</sup>
	280.0	1.73648177666930348289 <sup>-10</sup>	1	6.4 <sup>-19</sup>	1.3 <sup>-19</sup>
	319.0	7.54709580222771957017 <sup>-10</sup>	1	9.9 <sup>-19</sup>	2.2 <sup>-19</sup>
	360.0	9.9999999999999999997987 <sup>-10</sup>	1	2.0 <sup>-18</sup>	2.0 <sup>-18</sup>
	393.5	8.33885822067168181505 <sup>-10</sup>	1	2.9 <sup>-18</sup>	6.3 <sup>-19</sup>

```

Test(<<10.136>>);
begin integer m,i,il,k,kl,maxi,j;
real pi,pil,max,maxl,angle,anglel,arg,argl,standard,standardl,led,ledl,
sum,suml,pow,powl,argsq,argsql,maxarg,maxargl,maxstd,maxstdl;
  outtext(<< double arctan>>,outcr,outcr,outcr);
  il:=kl:=0;
  for j:=1,2 do
    begin
      assign(pi,doubleinone);
      outcr; doubleoutput(20,outtext(<<pi = >>,pi,outcr);
      outtext(outcr,<<degr. x>>,outsp(27),<<arctan(x)>>,outsp(20),
        <<arctan(x) - Taylor>>,outsp(11),<<arctan/pi*180>>,
        outsp(16),<<rel.error>>,outcr);
      for m:=100 step if m<200 then 100 else 500 until 5200 do
        begin
          assign(max,-1);
          for i:=m-(if m<200 then 100 else 500) step
            if m<200 then 1 else 10 until m do
            begin
              assign(angle,mult(pi,div(i,18000)));
              assign(arg,div(doublesin(angle),doublecos(angle)));
              if i<2700 then
                begin
                  assign(pow,minus(power(arg,3)));
                  assign(argsq,minus(mult(arg,arg)));
                  k:=3; assign(sum,0);
L:                assign(pow,mult(pow,argsq));
                  k:=k+2;
                  assign(led,div(pow,k));
                  assign(sum,add(sum,led));
                  if sub(doubleabs(led),mult(10-20,doubleabs(sum)))>0
                     then goto L;
                  assign(standard,add(mult(argsq,div(arg,3)),add(sum,arg)));
                end
              else assign(standard,angle);
              assign(angle,doublearctan(arg));
              if sub(doubleabs(sub(standard,angle)),max)>0 then
                begin
                  maxi:=i;
                  assign(maxarg,arg);
                  assign(maxstd,standard);
                  assign(max,doubleabs(sub(standard,angle)))
                end
              end;
              output(<<nd.dd>>,outcr,maxi/100,outsp(1));
              assign(angle,doublearctan(maxarg));
              doubleoutput(20,maxarg,outsp(1),angle,outsp(1),maxstd,outsp(1),
                mult(180,div(angle,pi)));
              output(<<-d.d10-dd>>,outsp(1),div(max,add(angle,10-21)))
            end;
          end;
        outchar(42)
      end;
    end;
  comment data:
  3.14159265358979323835, 3.14159265358979323855;

```

$$\pi = 3.14159265358979323833$$
[illegible]

January 1966

Double precision arithmetic in GIER ALGOL III

MACHINE CODES

[JZ 20.12.1965]

b k=c60+e70, i=10, a5, b2

[assign(a,b): ]

b:	ncn (c47)	t3	; if number of param=2
	ps 15	,hh c55	; then alarm(<<param>)
	ps (c46)	,pm s1	; M:=description(a)
	hs c22		; goto take formal
	pm c36	,ps (c46)	
	gm s1	,pm s2	; formal[1]:=UA; M:=description(b)
	grn 1c37	,hs c22	; UV1:=0; goto take formal
	ps (c46)	,ps (s1)	; s:=edr(a)
	pp (c36)	,arn p	; R:=b
	gr s	,pm p1	; a:=b; M:=b1
	gm s1	,ps (c46)	; a1:=b1; s:=last used
	qq (c46)	t1	; last used:=last used+1
	gr c37	,hv ra3	; UV:=0; goto exit

[double abs(a): ]

b1:	ncn (c47)	t2	; if number of param=1
	ps 15	,hh c55	; then alarm(<<param>)
	ps (c46)	,pm s1	; M:=description(a)
	grn 1c37	,hs c22	; UV1:=0; goto take formal
	ps (c36)	,arnf s	; R:=mantissa(head(a)); M:=exp(a)
	hv ra	LT	; if a<0 then goto minus
a4:	pm s	,gm c37	; E: UV:=a
	pm s1	,gm 1c37	; UV1:=b
	ps (c46)	,hv c51	; s:=last used; goto RS

[minus(a): ]

b2:	ncn (c47)	t2	; if number of param=1
	ps 15	,hh c55	; then alarm(<<param>)
	ps (c46)	,pm s1	; M:=description(a)
	grn 1c37	,hs c22	; UV1:=0; goto take formal
	ps (c36)	,arnf s	; R:=mantissa(head(a)); M:=exp(a)
	hh ra4 X	LZ	; if a=0 then goto E
a:	gr c68 X		; minus: save head of mantissa of a
	ga c47	,arn s1	; save exponent; R:=tail of a
	mt r2	,tl -39	; M:=-tail; R39:=mente
	sr c68	,nl ra1	; R:=mente-head of mantissa
	tl -10	,gr c37	; normalise float.; UV:=R
	arn c47	,ps (c46)	; s:=last used
a1:	ar 0 D	ITA t10	; calculate exponent of the result
	grn c37 XV	LO	; if exp<-512 v exp>511 then
	ga c37 V		; begin UV:=M:=0
	hv 0	MTA	; if exp>511 then alarm(<<spill>) end else
a3:	gm 1c37	,hv c51	; UV[0-9]:=exp; UV1:=M; goto RS

b k=e90, i=0

d i=e89, b1=b1-b, b2=b2-b

qq 144.9+c60.19

tassign;

qq 144.9+c60.19+b1.29

tdoubleabs;

qq 144.9+c60.19+b2.29

tminus;

d a5=e68+e68+e68+e68

d e71=e71+a5-e68, e72=e72+c68+e68

d e89=e89+a5+a5

e

d c60=c60+e68

e



[JZ 20.12.1965]

b k=c60+e70,i=10,b2

[add(a,b): ]

ncn (c47)	t3			; if number of parameters $\neq$ 2
ps 15	,hh	c55		; then alarm({<param>})
ps (c46)	,pm	s1		; M:=description(a)
grn 1c37	,hs	c22		; UV1:=0; goto take formal
pm (c36)	,ps	(c46)		
gm s1	,is	(c36)		; formal[1]:=a
pm s1	X			
pm s2	,gr	s2		; M:=description(b); formal[2]:=a1
grn 1c37	,hs	c22		; UV1:=0; goto take formal
ps (c46)	,pp	(c36)		; s:=addr(a)-1; p:=addr(b)
pp p-1				; p:=p-1
arnf p1	X	IZB		
ga c47	,gm	c54		; c47[0-9]:=expb; loc[c54]:=b
arnf s1	X	IZA		
sr (c47)D	ITB			; Raddr:=expa-expb
ga c47	LZB			; if b=0 then c47[0-9]:=expa
mt -1	D	MTB		; if expa<expb then Raddr:=Raddr*(-1)
ar 68	D	ITA		; Raddr:=Raddr+68; TA:=abs(expa-expb)>68
ga rb	,arn	s1		; b[0-9]:= -abs(expa-expb)+68
ps p	XV	LTB		; if expa>expb then begin x:=a; y:=b; x1:=a1
ga c47	V	NZC		; y1:=b1; if a $\neq$ 0 $\wedge$ b $\neq$ 0 then c47[0-9]:= expa
				; and else begin
pp (c46)	V	LTB		; x:=b;y:=a; x1:=b1; y1:=a1 end
arn c54	,gm	c54		; loc[c54]:=x
pm p2	,pp	(c47)		; M:=y1; R:=y; p:=resulting exponent (nearly)
qq	V	NTA		
tlh -39	V	NZC		; if TA $\wedge$ a $\neq$ 0 $\wedge$ b $\neq$ 0 then R:=M:=y:=y1:=0 else
b: tl	Q	NZC	t-68	; shift to same exponent
gr c68	X			
ar s2	,tl	-39		; M:=y1+x1
ar c54	,ar	c68		; R:=mentex+y
nl rb1	,tl	-10		; normalize float.
gr c37	,ps	(c46)		; UV:=head of the result; s:=last used
qq (c46)	t1			; last used:=last used+1
b1: arn pQ	D	NZ	t10	; if result $\neq$ 0 then calculate exponent
ga c37	V	NO		; if exp $\geq$ -512 $\wedge$ exp $\leq$ 512 then set exponent in UVaddr
grn c37	XV	LT		; else if exp $\leq$ 0 then UV:=M:=0
hv 0	LO			; if exp $\geq$ 511 then alarm({<spill>})
gm 1c37	,hv	c51		; UV1:=M; goto RS

b k=e90,i=0

d i=e89

qq 144.9+c60.19

t add;

d e71=e71+e68, e89=e89+e68+e68

e  
d c60=c60+e68

e

[JZ 20.12.1965]

b k=c60+e70,i=10,b2

[sub(a,b): ]

```
ncn (c47)          t3          ; if number of parameters=2
ps 15              ,hh c55      ; then alarm({<param>})
ps (c46)           ,pm s2       ; M:=description(b)
grn 1c37           ,hs c22      ; UV1:=0; goto take formal
ps (c36)           ,pm s1       ;
ps (c46)           ,gm s2       ; formal[2]:=b1
pm (c36) X         ;
pm s1              ,gr s1       ; M:=description(a); formal[1]:=b
grn 1c37           ,hs c22      ; UV1:=0; goto take formal
pp (c46)           ,ps (c36)    ; s:=addr(a); p:=addr(b)-1
arnf p1 X IZB      ;
ga c47             ,gm c68      ; c47[0-9]:=expb
ps s-1             ,srn p2      ;
tl -39             ,sr c68      ;
gr c54             ,gm p2       ; loc[c54]:= -b; M:= -b
arnf s1 X IZA      ;
sr (c47)D ITB      ; Raddr:=expa-expb
ga c47             LZB         ; if b=0 then c47[0-9]:=expa
mt -1 D NTB        ; if expa<expb then Raddr:=Raddr*(-1)
ar 68 D ITA        ; Raddr:=Raddr+68; TA:=abs(expa-expb)>68
ga rb              ,arn s1      ; b[0-9]:= -abs(expa-expb)+68
pp s XV LTB        ; if expa>expb then begin x:=a; y:=-b; x1:=a1
ga c47 V NZC        ; y1:=-b1; if a+0^b=0 then c47[0-9]:= expa
                    ; end else begin
                    ; x:=-b;y:=a; x1:=-b1; y1:=a1 end
                    ; loc[c54]:=x
                    ; M:=y1; R:=y; p:=resulting exponent (nearly)
                    ; if TA^a+0^b=0 then R:=M:=y:=y1:=0 else
                    ;
                    ; shift to same exponent
                    ;
                    ; M:=y1+x1
                    ; R:=mente+x+y
                    ; normalize float.
                    ; UV:=head of the result; s:=last used
                    ; last used:=last used+1
                    ; if result=0 then calculate exponent
                    ; if exp>-512^exp<512 then set exponent in UVaddr
                    ; else if exp<0 then UV:=M:=0
                    ; if exp>511 then alarm({<spill>})
                    ; UV1:=M; goto RS

b:  tl Q           NZC t-68
    gr c68 X
    ar s2          ,tl -39
    ar c54         ,ar c68
    nl rb1         ,tl -10
    gr c37         ,ps (c46)
    qq (c46)       t1
b1:  arn pQ D NZ t10
    ga c37 V NO
    grn c37 XV LT
    hv 0 LO
    gm 1c37 ,hv c51
```

b k=e90,i=0

d i=e89

qq 144.9+c60.19

t sub;

d e71=e71+c68, e89=e89+e68+e68

e  
d c60=c60+e68

e

[JZ 20.12.1965]

b k=c60+e70,i=10,a2

[mult(a,b):]

```

ncn (c47)          t3
ps 15             ,hh c55
ps (c46)          ,pm s1
grn 1c37          ,hs c22
pm (c36)          ,ps (c46)
gm s1             ,is (c36)
pm s1             X
pm s2             ,gr s2
grn 1c37          ,hs c22
ps (c46)          ,pp (c36)
qq (c46)          t1
arnf p            X IZA
hv ra1            LZA
ga ra2            X
pm p1             ,tl 9
gr c37            ,gm 1c37
arnf s1           X IZA
ga c47            X V NZA
a1: grn c37        ,hv ra3
pm s2             ,tl 9
gr s2             ,mkn c37
pm s2             ,mk 1c37
ml c37            ,nl ra
tl -10            ,gr c37
arn c47
a2: ar Q          D
a:  ar Q          D NO t3
ga c37            V NO
hv 0              NT
grn c37           X LO
a3: gm 1c37        ,hv c51

```

```

; if number of parameters#2 then
; alarm(<<param>)
; s:=last used; M:=descr(a)
; UV1:=0; goto take formal
; s:=last used;
; formal[1]:=a;
;
; M:=descr(b); formal[2]:=a1
; UV:=0; goto take formal
; s:=last used
; last used:=last used+1
;
; if b=0 then goto zero
; save exponent(b)
;
; (UV,UV1) := mantissa(b,b1)
;
; if a=0 then save exponent(a) else
; zero: goto exit
;
; RM := mantissa(a,a1); save a
;
; RM:=(a,a1)*(b,b1); normalize the result
; UV:=head of mantissa(result)
;
;
; if exp>-512^exp<511 then UVaddr:=exponent
;
; else if exp>511 then alarm(<<spill>)
; exit: if a=0^b=0^exp<-512 then UV:=M:=0
; UV1:=M; goto R5

```

b k=e90,i=0

d i=e89

qq 144.9+c60.19

tmult;

d e71=e71+e68, e89=e89+e68+e68

e

d c60=c60+e68

e

[JZ 20.12.1965]

b k=c60+e70, i=10, a2

[div(a,b):]

```

ncn (c47)          t3
ps 15             ,hh c55
ps (c46)          ,pm s1
grn 1c37          ,hs c22
pm (c36)          ,ps (c46)
gm s1             ,is (c36)
pm s1             X
pm s2             ,gr s2
grn 1c37          ,hs c22
ps (c46)          ,pp (c36)
qq (c46)          t1
arnf p            X IZB
hv 0              LZB
ga ra             X
pm p1             ,t1 10
gr c54            ,gm c68
arnf s1           X IZA
ga c47            X V NZA
grn c37           ,hv ra2
pm s2             ,t1 8
gr s1             ,gm s2
dl c54            ,gr c37
mt -1             D X
mkn c68           ,ar s2
ml c54            ,ar s1
dl c54            ,ar c37
gr c37            X
dk c54            X
arn c37           ,nl ra1
t1 -10            ,gr c37
arn c47
a: sr Q           D
a1: ar Q           D NO t1
    ga c37         V NO
    hv 0            NT
    grn c37         X LO
a2: gm 1c37        ,hv c51

```

```

; if number of parameters=2
; then alarm({<param>})
; s:=last used; M:=descr(a)
; UV1:=0; goto take formal
; s:=last used
; formal[1]:=a
;
; M:=descr(b); formal[2]:=a1
; UV1:=0; goto take formal
; s:=last used; p:=adr(b)
; last used:=last used+1
;
; if divisor=0 then alarm({<spill>})
; save exponent(b)
;
; (c54,c68):=mantissa(b,b1)
;
; if a=0 then goto exit
; else save exponent(a)
;
; (formal[1],formal[2]):=mantissa(a,a1)
; UV:=c:=a/b
; M:=-c
; R:=a1-cxb1
; RM:=a-cxb+21(-39)x(a1-cxb1)
; R:=q1; M:=remainder
; UV:=q1; R:=remainder
; M:=q2x21(-39)
; normalize the quotient
; UV:=head of the mantissa(a/b)
;
;
; Raddr:=expa-expb+norm.exp.+1
; if exp>-512^exp<511 then UVaddr:=exp
; else if exp>511 then alarm({<spill>})
; if a=0^exp<-512 then UV:=M:=0
; exit: UV1:=M; goto RS

```

b k=e90,i=0

d i=e89

qq 144.9+c60.19

tdiv;

d e71=e71+e68, e89=e89+e68+e68

d c60=c60+e68

[JZ 20.12.1965]

b k=c60+e70,i=10,a4,b4

[power(a,n), n≥0]

```
ps (c46) ,it 3
ncn (c47) ,hv rb
pm s2 ,hs c22
arnf(c36) ,nc 0
b: ps 15 ,hh c55
tkf -29 ,ps (c46)
gr s2 ,pm s1
grn 1c37 ,hs c22
ps (c46) ,pp (c36)
pm s2 ,gm c54
arnf p ,gmf c47
pm p1 ,tl 9
gr s1 ,gm s2
grn c37 ,gr c36
grn 1c37 ,ud c51
pa c37 IOA t128
it sc37-1 ,pt rb1
b4: hv 0 LOA
arn c54 ,gs rb1
hv rb3 X NZ
arn c37 ,pm 1c37
tl -9 ,gr c37
gm 1c37 ,pp (c36)
gp c37 ,hh c15
b3: cln -1 ,gm c54
b1: nt Q[s or c37-1],pp Q[c37+s-1]
b2: nt c47[or c36],qq c47+c36
hv ra1 LZ
pm s2 ,mkn p1
pm s1 ,mk p2
ml p1 ,nl ra
tl -1 ,gr p1
arn (rb2) ,ar c47
a: ar Q D NO t3
hv ra2 NO
pi 1.0 NT
xrn ,grn p1
a2: ga (rb2) ,gm p2
a1: pm r X
```

d a3=b4-i, a4=b1+a3

nt a3[or b1-i],hv ra4

b k=e90,i=0

i=e89

qq 144.9+c60.19

tpower;

d e71=e71+e68, e89=e89+e68+e68

e  
d c60=c60+e68

e

```
; s:=lastused; if number of parameters ≠ 2
; then goto error
; M:=description(n); goto take formal
; if n<0 then
; error: alarm(<<param>)
; formal[2]:=n; comment rounded to integer
; M:=description(a)
; UV1:=0; goto take formal
;
; rest of n := n
; loc[c47]:=exponentpart of a
;
; (formal[1],formal[2]):=mantissa of a
; UV:=UA:=0
; UV1:=0; lastused:=lastused+2
; result:=1; float overflow:=false
;
; next bit: if float overflow then alarm
; R:=rest of n; first:=true
; if rest of n = 0 then
; begin comment finished;
;
; (UV,UV1):=result
; goto Running System end;
; take next bit of rest of n
; second:p:=if first then c37-1 else last used
; addr(b2):=if first then c36 else c47
; if last bit of rest of n ≠ 0 ∧ first
; then result:=result×a
; else a:=a/2
;
; calculate exponent part of if first then
; result else a
; if exponent part>511 ∨ exponent part <-512
; then begin float overflow:=true else
; exponent part:=product:=0 end
;
; comment R:=loc[r] => R≠0
;
; relative addresses
;
; if first then begin first:=-,first;
; goto second end else goto next bit
```

[JZ 20.12.1965]

b k=c60+e70,i=10,a10

[double sqrt(a): ]

```

ncn (c47)      t2
ps  15         ,hh c55
ps  (c46)      ,pm s1
grn 1c37       ,hs c22
ps  (c36)      ,pm s1
gm  1c37       ,arnf s
ps  8
hh  c55        LT
hv  ra X       LZ

pa  ra2 X
ar  2 D
tk  -1         ,ga ra1
tk  10         ,ck -9
ga  ra6 X
pm  1c37       ,it 8
a6: tl Q        ,gr ra3
gm  ra4        ,tl 1
gr  c37        ,gm 1c37
a2: bt Q        t940
nl  ra7        ,hh ra8
arn ra3        ,pm ra4
dl  c37        ,gr c68
mt  -1 D X
mkn 1c37       ,ar ra4
ml  c37        ,ar ra3
dl  c37        ,ac c68
xr  0          ,dk c37
pm  1c37       ,gr 1c37
arn  c37       ,tl -1
gr  c37 X
ar  1c37       ,tl -39
ar  c68        ,ar c37
a8: hv ra2-1    ,tl -10
a7: it Q        t-1
a1: ar Q        D

a:  gr c37      ,gm 1c37
ps  (c46)      ,hv c51
a3: qq [full word]
a4: qq [full word]

```

```

b k=e90,i=0
d i=e89
    qq 144.9+c60.19
    tdoublesqrt;
    d e71=e68+e71, e72=e72+e68
    d e89=e89+e68+e68+e68
    e
d c60=c60+e68

```

e

```

; if number of parameters=1
; then alarm({<param>})
; s:=last used; M:=descr(a)
; UV1:=0; goto take formal
; s:=UA
; UV1:=x2; RF:=x1x2,u
; s:=text nr({<sqrt>})
; if a<0 then alarm({<sqrt>})
; if x=0 then
; begin
;   Radr:=u; M:=x1; i:=0
;
;   v:=(u+2)/2
;   m:=if u even then 0
;       else 1
;
;   (x1,x2):=(x1,x2)x2^(8+m)
;   (y01,y02):=RM:=(x1,x2)x2
; new: (yi1,yi2):=RM
;   i:=i+1; if i=7 then
;   begin normalize yn; goto exit end
;   RM:=(x1,x2)
;   The following piece
;   of code is
;   division of two double
;   precision fixed point
;   numbers and after that
;   addition of two double precision
;   numbers
;
;   RM:=(yi+a/yi)/2
;   goto new; exit:
;
;   form exponent
; end
; store result
; goto Running System
; working location
; working location

```

[IM 22.12.1965]

b k=c60+e70,i=10,a20,b20

[double sin(y), track 1]

```

a1:  ncn (c47)      t2      ; if number of parameters ≠ 1 then
      ps 15      ,hh c55    ; then alarm({<param>})
      ps (c46)    ,pm s1    ; M:=description(y)
      grn 1c37    ,hs c22    ; UV1:=0; goto take formal
      arnf(c36) X ITA      ; TA:=y<0; M:=mantissa(head(y))*2-11
      ga ra9      ,it (ra9)  ; a9[0,9]:=expy
      bs -508     ,hvn ra10   ; if expy < -508 then begin R:=0; goto mod end;
      arn (c36) X t1      ; RM:=mantissa(y)*2-11
      tl 9        ,hs ra8    ; RM:=RM*2-9*1/pi
      nl r1       ,gi ra6    ; save indicator
      it Q        t2      ;
a9:   nt Q        ,bs 0      ; if expy+normexp+2>0 then
a10:  tl (ra9)    ,pa ra9    ; mod: begin RM:=(y/pi(mod 1))*2; a9[0,9]:=0;
[s5]  qq ra       ,tl 1      ; RM:=RM/2
      tl -2       ,gr c37    ; R00:=R0; UV:=R
      gm 1c37     ,tl (ra9)  ; UV1:=M; RM:=RM*2-1(a9[0,9])
      gr c54      ,mkn c54   ;
      tk 1        ,pm c54    ;
      ml c54      ,gm c54    ; RM:=RM1/2
      sr 64       D        ; RM:=RM-1/8
      hs c65      ,qq 1c60.29+1 ; call next track
      gr c68      ,hs s      ;

```

[The content of RM is in this moment  $K/16$ , where  $K=16 \times ((z/2)^{1/2} - 1/8)$ , where  $z=2 \times (\text{if } x < 1/2 \text{ then } x \text{ else } x-1)$ , and  $x=y/2/\pi \pmod{1}$ . goto next track, Chebysev evaluation]

```

a:    162/998/ 54/915      ; head and
      290/ 42/335/901     ; tail of 1/pi

a2:   gr c54 X            ; Return to here from Chebysev evaluation
      sr sb3      ,tl -39 ;
      ar c54      ,sr sb7 ;
a6:   pi Q[TA]    ,hs ra8  ;
      nl ra4      ,ck 0[R00:=0] ;
      tl -10     ,pp (ra9) ;
a4:   AR P1/2 D NZ t3    ;
      ps (c46)    ,gm 1c37 ;
[s5]  gr c37      ,hv c51  ;
a8:   gr c54      ,pp (s5) ;
      grn c36     ,mkn p    ; UA:=0
      pm c54      ,mk p1   ;
      ml p        ,ud ra2  ;
      hr s1 X NTA      ;
      mt ra4-1    ,tl -39  ;
      sr c54      ,hr s1   ;

```

[double sin(y), track 2]

d i=a1+40

```
b9:  it  rb5      ,pa  rb6      ;
      pp  1       ,gm  rb12     ;
b13:  pm  rb5      ,gm  c53      ;
      pm  rb12     ,gm  rb10     ;
      pm  c36      ,gm  rb11     ;
      gr  rb12     ,pm  c53      ;
      gm  c36      ,min c68      ;
      pm  rb12     ,mk  c54      ;
      ml  c68      ,tl  4       ;
      gr  c53 X    ;
b6:   ar  p[b5]    ,sr  rb11     ;
      tl  -39      ,ar  c53      ;
      sr  rb10     ,bs  p-4     ;
      pp  p1       ,ar  (rb6)    ;
b:    bs  p-18     ,hs  s3      ;
      pp  p1       ,hh  rb13     ;
```

```
b5:   qq  377.39   ;
      1023/1023/ 940/ 258   -65 ;
      0/ 16/ 655/ 830   +88 ;
      1021/ 268/ 444/ 921   -86 ;
      374/ 805/ 908/ 690   -6 ;
      276/ 659/ 244/ 891   +27 ;
      1023/1023/1023/ 944   ;
      361/ 285/ 190/ 227   -69 ;
      0/ 0/ 6/ 487   ;
      149/ 428/ 325/ 11   -75 ;
      1023/1023/ 624/ 942   ;
      11/ 456/ 640/ 760   -13 ;
      0/ 16/ 792/ 776   ;
      395/ 716/ 233/ 398   +20 ;
      1023/ 558/ 940/ 98   ;
      222/ 485/ 285/ 121   +15 ;
      7/ 131/ 339/ 209   ;
      159/ 902/ 778/ 256   -32 ;
      974/ 193/ 632/ 540   ;
      198/ 190/ 993/ 157   +56 ;
      86/ 247/ 496/ 192   ;
```

```
b10:  qq  [working location] ;
b11:  qq  [working location] ;
b12:  qq  [working location] ;
```

1070 1071



[double cos(y): ]

d i=a1+80

```

ncn (c47)          t2          ; if number of parameters $\neq$ 1 then
ps 15              ,hh c55      ; alarm(<<param>)
ps (c46)           ,pm s1       ; M:=description(y)
grn 1c37           ,hs c22      ; UV1:=0; goto take formal
arnf(c36) X        ;
ga ra3             ,it (ra3)    ;
bs -508            ,hvn ra3     ;
arn (c36) X        t1          ;
tl 9               ,hs ra5      ;
nl c47             ,it (c47)    ;
a3: tl             ,tl 2        ;
ar 256 D           ;
tl 1               ,tl -2       ;
qo ra11            ,gm 1c37     ;
gr c37             ,mkn c37     ;
pm c37             ,tk 1        ;
ml c37             ,gm c54      ;
sr 64 D            ;
hs c65             ,qq 1c60.29+1; call track with Chebysev evaluation
gr c68             ,hs s        ;

```

```

a11: 162/998/ 54/915          ; head and
      290/ 42/335/901         ; tail of 1/pi

```

```

a7: gr c54 X                ;
sr sb3                     ,tl -39 ;
ar c54                     ,sr sb7 ;
a14: hs ra5                 ;
nl ra12                    ,ck     ;
tl -10                     ;
a12: ar 0 D NZ t3           ;
ps (c46)                   ,gm 1c37 ;
gr c37                     ,hv c51 ;
a5: gr c54                  ,pp (s5) ;
grn c36                    ,mkn p   ;
pm c54                     ,mk p1  ;
ml p                       ,hr s1   ;

```

d b3=b11-b, b7=b3-1

b k=e90,i=0

d i=e89

qq 144.9+c60.19

tdoublesin;

qq 144.9+2c60.19

tdoublecos;

d e71=e71+e68+e68,e72=e72+e68+e68,e89=e89+e68+e68+e68+e68+e68+e68

e

d c60=c60+e68+e68+e68

le

[IM 22.12.1965]

b k=c60+e70,i=10,a20,b20

```
[double arctan(y)
  y=z*21/4exp = z*21/4(fl.exp+2), (1/4≤z<1/2)
  y:= if abs(y)≤ 1 then y else 1/y;
  if abs(y)>1 then
    y = 1/z*21/4(-fl.exp - 2) = 1/16/z*21/4(-abs(fl.exp)+2) ]
```

[track 1.]

a1:	ncn (c47)	t 2	;if number of param. ≠ 1 then
	ps 15	, hh c55	;alarm ({<param>})
	ps (c46)	, pm s1	;M:= descr.(y)
	grn 1c37	, hs c22	;UV1:= 0 , goto taka formal
	arn (c36)	ITA	;TA:= fl.exp<0
	ga c47	, smn c47	;c47[0-9]:= -abs(fl.exp)
	ga c47	, ps (c36)	
	arnf s	ITB	;TB:= y<0
	hv ra	LZ	;if y=0 then goto a
	pm s1	, tl 9	;RM:= z:= numerus(y)*2 <sup>1/4</sup> (-1)
	hh ra2	LTA	;if fl.exp<0 then goto a2
	gr c37	, gm 1c37	;else begin c37,1c37:= z1,z2
	pmn 32	DX	;RM:= 1/16
	dl c37	, gr c68	;c68:= c:= RM/z1
	mt -1	DX	;M:= -c
	mlkn 1c37	, ml c37	;RM:= -c*(z1+exz2)
	ar 32	D	; +1/16
	dl c37	, ar c68	
	gr c68	X	
	dk c37	X	;RM:= z:= RM/z + c
a2:	arn c68	, gr c37	;c37,1c37:= z
	gm 1c37	, tl (c47)	;RM:= y/4:= z*2 <sup>1/4</sup> (-abs(fl.exp))
	gr c68	, grn c36	;c36:= 0
	mlkn c68	, tk 1	;RM:= K/64:= (y <sup>1/2</sup> -1/2)/16
	pm c68	, ml c68	
	sr 16	DIK	;save indicator
	gr c68	, gm c54	;c68,c54:= K/64
	hs c2	, qq 1c60.29	;goto next track
a:	pi 1.2	, hh ra2	;TA:= 1
	qq		;free

[track 2. doublearctan(y)]

i=a1+40

```
b9:  hs  c65      , qq 2c60.29+1 ;take constant track
      gs  rb6      IK           ;restore indicator
      grm rb12     , pp  1      ;initializing
b8:  pm  s         , gm  c53     ;M:= c53:= t32
      pm  rb12     , gm  sb10    ;t11:= t21
      pm  c36      , gm  sb11    ;t12:= t22
      gr  rb12     , pm  c53     ;t21:= t31 , t22:= M:= t32
      gm  c36      , mkn c68    ;RM:= Kxt2
      pm  rb12     , mk  c54
      ml  c68      , tl  6
      gr  c53      X
b6:  ar  p[s]      , sr  sb11    ;t3:= Kxt2 - t1 + C[p]
      tl  -39      , ar  c53
      sr  sb10     , bs  p-11
      pp  p1       , ar  (rb6)  ;if p>11 then double-cell-const.
      bs  p-36     , hv  r2
      pp  p1       , hh  rb8
      sr  sb10     , ud  rb6-1  ;t3:= t3-t1
      sr  sb11     , tl  -39    ;arctan:= zxt3
      ac  c53      , mkn c37
      pm  c53      , mk  1c37
      ml  c37      , pp  (c47)
      hv  ra3      LTA         ;if fl.exp<0vy=0 then goto a3
      tl  p        , gr  c68   ;else
      gm  c54      , arm ra7   ; arctan:= sign(y)*pi/2-arctan
      mt  ra8      LTB
      sr  c54      , tl  -39
      sr  ra5      LTBV
      ar  ra5
      sr  c68      , pp
a3:  nl  ra4      , ck         ;normalize
a8:  tl  -10      , ps (c46)   ;shift to fl.p. position
a4:  ar  p        DNZ t 4      ;if R≠ 0 then R:= exponent
      tln -39     LO          ;if exp<-512 then RM:= 0
      gr  c37      , gm  1c37  ;UV,UV1:= doublearctan
      grm c36      , hv  c51   ;c36:= 0, exit
a5:  25/135/949/273
a7:  22/560/564/787
b12: qq          ;pi/32
                        ;working location: t21
```

[track 3. doublearctan(y)]

i=a1+80

a10:

0/	0/	0/
1023/	1023/	1022/
0/	0/	11/
1023/	1023/	952/
0/	0/	435/
1023/	1021/	404/
0/	15/	1018/
1023/	925/	515/
0/	608/	505/
1020/	323/	254/
22/	959/	309/
880/	552/	416/
390/	124/	283/
0/	0/	0/
435/	588/	841/
1023/	1023/	1023/
37/	551/	147/
0/	0/	0/
460/	403/	551/
1023/	1023/	1023/
374/	701/	102/
0/	0/	2/
381/	681/	569/
1023/	1023/	1004/
418/	668/	457/
0/	0/	128/
368/	812/	1023/
1023/	1023/	144/
494/	367/	736/
0/	6/	88/
21/	45/	32/
1023/	978/	758/
342/	301/	783/
0/	364/	920/
283/	45/	865/
1020/	626/	103/
468/	155/	193/
56/	417/	716/

377 ;c[0]
159 ;c[1]
636 ;c[2]
815 ;c[3]
540 ;c[4]
615 ;c[5]
695 ;c[6]
676 ;c[7]
1019 ;c[8]
458 ;c[9]
568 ;c[10]
927 ;c[11]
440 ;c[12]
1
732 ;c[13]
1012
960 ;c[14]
71
460 ;c[15]
566
1015 ;c[16]
923
389 ;c[17]
897
603 ;c[18]
217
898 ;c[19]
372
438 ;c[20]
511
753 ;c[21]
805
511 ;c[22]
875
75 ;c[23]
65
802 ;c[24]
188

C(84)	1023 / 1023
C(52)	0 / 9
C(48)	1023 / 971

m=49

m=59

224	324
184	81
547	707
844	739
425	535
701	651
538	648
786	676
838	988
476	366
444	604
417/ 6	921
252	362
770	660
787	897
477	267
874	983
568	485
552	662
1018	908
306	415
763	665
566	585
864/ 366	865/ 104
147/ 41	145/ 712

b: qq  
b1: qq

;working location: t11  
; t12

db10=b-a10,b11=b1-a10

b k=e90,i=0

d i=e89

qq 144.9+c60.19

tdoublearctan;

d e71=e71+e68,e72=e72+e68,e89=e89+e68+e68+e68

e

dc60=c60+e68+e68+e68

e

[PVV 21.12.65]

[double ln(x)]

b k=c60+e70,i=10,a6,b10

[track 1]

d b=40i

	ncn (c47)	t	2	; if number of parameters=1
	ps 15	,hh	c55	; then alarm({<param>})
	ps (c46)	,pm	s1	; M:=description(x)
	grn 1c37	,hs	c22	; UV1:=0; goto take formal
	arnf(c36)	,gm	c47	; RF:=x[0]; c47:=exponent(x)
	hv ra3	LZ		; if x=0 then goto error
	pm (c36)	t	1 V NT	; if x>0 then M:=x[1]; skip next
a3:	ps 5	,hh	c55	; error: alarm({<ln>})
	hs c65	,qq	1c60.29+1	; take constant track
	sr c40	,tl	1	; t:=(numerus(x)-1)*2
	sr c40	,tl	11	; -1
	gr c37	,gm	1c37	
	grn c68	,gr	c54	; b1:=0; comment b1: c68,c54
	pm sb5	,gs	ra2	; b:=a[n]; comment b: R,M
a1:	pp b5	,pp	p-1	; p:=number of constant cells
				; again: p:=p-1
	gm c36	,pm	c68	; save b[1]
	gm ra5	,pm	c54	; b2:=b1; comment b2: ra5,sb9
	gm sb9	,pm	c36	
	gr c68	,gm	c54	; b1:=b
	mkn c37	,pm	c68	; b:=b1*t
	mk 1c37	,ml	c37	
	tl 1	,sr	ra5	; x2 - b2
a4:	gr c36	X		
a2:	ar p	,sr	sb9	; + a[p]
	tl -39	,ar	c36	
	bs pb8	,hv	ra6	
	pp p-1	,ar	(ra2)	; if double cell constant
a6:	bs p	,hh	ra1	; if p>0 then goto again
	sr ra5	,ud	ra4	; b:=(b-b2)/2
	sr sb9	,tl	-39	
	ar c36	,ar	c47	; double ln:=(b+exponent(x))
	gr c68	,grn	c36	
	mkn sb6	,pm	c68	; x ln(2)
	mk sb7	,ml	sb6	
	tl 1			
	nl ra	,tl	-10	; normalize; shift to fl.p. position
	gm 1c37	,ps	(c46)	
a:	pm	t	8 D	; exponent(double ln)
	grf c37	,hv	c51	; exit
a5:	0			; working location b2[0]

b k=e90,i=0

d i=e89

qq 144.9+c60.19

; pass 6 word

tdoubleln;

d e71=c71+e68, e72=e72+e68, e89=e89+c68+e68+e68

e

d c60=c60+e68+e68

[double ln(x) track 2]

b:	0/ 556/ 144/ 566	; constantsx2,(-10): a0
	53/ 381/ 566/ 463	
	0/ 253/ 479/ 229	; a1
	279/1016/ 744/ 971	
	1023/1002/ 262/ 25	; a2
	358/ 69/ 715/ 6	
	0/ 2/ 498/ 844	; a3
	351/ 747/ 646/ 94	
	1023/1023/ 696/ 282	; a4
	30/ 187/ 831/ 56	
	0/ 0/ 144/1006	; a5
	258/ 507/ 538/ 494	
	1023/1023/1017/ 582	; a6
	52/ 752/ 508/ 222	
	0/ 0/ 0/ 968	; a7
	275/ 594/ 951/ 406	
	1023/1023/1023/ 878	; a8
	305/ 664/ 256/ 8	
	0/ 0/ 0/ 22	; a9
	89/ 773/ 283/1007	
	1023/1023/1023/1020	; a10
b10:	294/ 822/ 514/ 497	
	273/ 466/ 405/ 750	; a11
	980/1016/ 70/ 409	; a12
	6/ 830/ 842/ 171	; a13
	1022/ 936/ 806/ 455	; a14
	0/ 177/ 966/ 310	; a15
	1023/ 995/ 386/ 929	; a16
	0/ 4/ 636/ 863	; a17
	1023/1023/ 257/ 87	; a18
	0/ 0/ 124/ 672	; a19
	1023/1023/1003/ 698	; a20
	0/ 0/ 3/ 328	; a21
	1023/1023/1023/ 467	; a22
	0/ 0/ 0/ 91	; a23
	1023/1023/1023/1009	; a24
b1:	0/ 0/ 0/ 3	; a25
b2:	177/ 456/ 383/ 500	; ln(2)
	231/ 755/ 350/ 316	
b4:	0	; working location b2[1]

d b3=1b2, b5=b1-b, b6=b2-b, b7=b3-b, b9=b4-b, b8=b-b10

e

[PVV 23.12.65]

[double exp(x)]

b k=c60+e70,i=10,a5,b15

[track 1]

d b=40i

ncn (c47)	t	2		; if number of parameters=1
ps 15	,hh	c55		; then alarm({<param>})
ps (c46)	,pm	s1		; M:=description(x)
grn 1c37	,hs	c22		; UV1:=0; goto take formal
annf(c36)	,srf	ra4		; if abs(x)>511*ln(2)
arnf(c36)	V LT			
ps 9	,hv	ra		; then goto large
gm c47	,gr	c68		; c47:=exponent(x)*2 <sup>1</sup> (-9)
				; c68:=numerus(x[0])*2 <sup>1</sup> (-11)
				; M:=x[1]
				; t:=x/ln(2)
pm (c36)	t	1		
mkn ra1	,pm	c68		
mk ra2	,ml	ra1		
gr c37	,gm	1c37		
tl (c47)	t	3		
ga c36	,arn	c37		; v:=entier(t)
pm 1c37	,it	10		
tl (c47)	,cl	-1		; t:=(t-v
sr 256	D			; -0.5)
tl 1	,gr	c37		; x 2
gm 1c37	,pp	(c36)		
hs c65	,qq	1c60.29+1		; take constant track
grn c68	,gr	c54		; b1:=0; comment b1: c68,c54
pm sb5	,gp	c36		; b0:=a[n]; comment b0: R,M
gs sb7				
hs sb6				; goto Chebyshev recurrence
sr sb12	,ud	sb14		; double exp:=(b0-b2)/2
sr sb13	,tl	-39		
ar c47	,nl	ra3		; normalize
a3: it 0	,it	1		; correct v
qq (c36)	,tl	-10		; shift to fl.p. position
gm 1c37	,pm	c36		; x2 <sup>1</sup> v
grf c37	,grn	c47		
a5: ps (c46)	,hv	c51		; exit: return to RS
a: grn 1c37	,arf	(c36)		; large:
hh c55	NT			; if x>0 then alarm({<exp>})
grn c37	,hv	ra5		; else double exp:=0; goto exit
<u>f</u>				
a4: 354.1982				; 511*ln(2)
<u>m</u>				
a1: 369/ 337/ 869/ 174				; log base2 (e)/2
a2: 23/ 962/ 955/ 930				
qq				; free space
qq				

[double exp(x) track 2]

```

b:      186/ 507/ 908/ 942      ; constants*2*(-2): a0
        315/ 627/ 681/ 262
        31/ 861/ 808/ 707      ; a1
        41/ 554/ 854/ 107
        2/ 763/ 44/ 19        ; a2
        260/ 849/ 446/ 28
        0/ 161/ 990/ 754      ; a3
        112/ 869/ 304/ 426
        0/ 7/ 6/ 356          ; a4
        192/ 527/ 196/ 842
        0/ 0/ 248/ 404        ; a5
        509/ 405/ 252/ 697
        0/ 0/ 7/ 172          ; a6
        446/ 716/ 826/ 658
        0/ 0/ 0/ 181          ; a7
        321/ 458/ 951/ 214
        0/ 0/ 0/ 3            ; a8
b10:    477/ 489/ 79/ 189      ; a9
        38/ 772/ 883/ 667
        0/ 687/ 510/ 619      ; a10
        0/ 10/ 847/ 831       ; a11
        0/ 0/ 160/ 85         ; a12
b1:      0/ 0/ 2/ 137          ; a13
                                ; Chebyshev recurrence:
b2:      pp      b5      ,pp    p-1
                                ; p:=number of constant cells
                                ; again: p:=p-1
                                ; save b0[1]; b2:=b1; comment b2: rb3,rb4
        gm      c47      ,pm    c68
        gm      rb3      ,pm    c54
        gm      rb4      ,pm    c47
        gr      c68      ,gm    c54      ; b1:=b0
        mk      c37      ,pm    c68      ; b0:=b1xt
        mk      1c37     ,ml    c37
        tl      1        ,sr    rb3      ; x2 - b2
b15:    gr      c47      X
b8:      ar      p        ,sr    rb4
        tl      -39      ,ar    c47
        bs      pb9      ,hv    rb11
        pp      p-1      ,ar    (rb8)    ; if double cell constant
b11:    bs      p        ,hh    rb2      ; if p>0 then goto again
        hr      s1
b3:      0
b4:      0                  ; return to track 1
                                ; working locations: b2[0]
                                ; b2[1]

```

d b5=b1-b, b6=b2-b, b14=b15-b, b7=b8-b, b9=b-b10, b12=b3-b, b13=b4-b

```

b k=e90,i=0
d i=e89
    qq 144.9+c60.19          ; pass 6 word
    tdoubleexp;
d e71=e71+e68, e72=e72+e68, e89=e89+e68+e68+e68
e

```

d c60=c60+e68+e68

e



[TA 22.12.65]

b k=c60+e70, i=10, a40

[doubleinput, doubleinone, doubletypein]

```
a:   sm (c47) DX t1 ; appetite+1
     ps (c46) ,hv ra2 ; s:=last used
a1:  ps (c46) t1 ; s:=last used:=last used +1
     pm s-1 IRC ; return information
     arn (s) DIZA t-1 ; count parameters
     gm s X MRC ; move return information
     hv c15 LZA ; exit if no more parameters
a2:  arn s1 ,gm s1 ; next parameter, save counter
     hh ra4 X LA ; expression
     ga c36 tk 10 ;
     ca 0 ,hvn ra3 ; simple variable
     ar (c36) t-1 ; array length
     ga c36 ,tk 30 ; UA:=addr.first element
     sc c36 ;
     sr 2 D ; R:=array length-2
a3:  pm ra9 ,ga c47 ; M:=return1, set number of numbers
     hs c65 ,qq 1c60.29+1; call double number read 1
     gm sa25 ,gp sa24 ; set return, save p
a4h: hv s1 ,hs c22 ; goto read, expression in input
     hvn ra3 ;
a5:  pm ra10 ,hh ra7 ; M:=return 2
a6:  pm ra11 ,arn c63 ; M:=return 3, save sum
     gr c68 ud c90 ; select type
     it (c48) ,pa c82 ; save case
a7h: pa c48 ,ud c12 ; case :=0, set UA
a8h: hhn ra3 ,pm c68 ; reset sum
     gm c63 ,it (c82) ; reset case
     pa c48 ,hv c12 ; return from typein
d a1=a1-a, a8=a8-a
a9:  hs c2 ,qq c60.29+a1; exit from read after doubleinput
a10: hv c12 ; exit from read after doubleinone
a11: hsf c2 ,qq c60.29+a8; exit from read after doubletypein
     qq ; empty
     qq ; empty
     qq ; empty
     qq ; empty
     qq ; empty
     qq ; empty
     qq ; empty
     qq ; empty
     qq ; empty
```

d a5=a5-a, a6=a6-a

b k=e90, i=0

d i=e89

qq 192.9+c60.19

tdoubleinput;

qq 190.9+c60.19+a5.29+91.39f

tdoubleinone;

qq 190.9+c60.19+a6.29+91.39f

tdoubletypein;

d e71=e71+e68+e68+e68, e72=e72+e68+e68+e68

d e89=e89+e68+e68+e68+e68+e68+e68+e68+e68+e68

e

[double number read, first track]

```

d a41=12e0+e96
a12: qq (c36) V t2 ; UA:=UA+2
      hs c65 ,qq a41.29+1; call basic read
      pp ,grn c53 ; state:=0, exp1:=0
      grn c37 ,grn 1c37 ; number:=0
a13h: pa c84 ,grn c54 ; sign:=plus, exp2:=0
d a26=7 [a15-a14], a27=9 [a16-a14]
  a28=23 [a19-a14], a29=27 [a20-a14]
a14: hsn (c91) IZA ; next: goto basic read
      qq a29.6+ 6.12+436.21+272.30+ 5.39; terminator parameter
      qq a28.6+26.12+ 65.21+ 31.30+ 71.39; digit -
      qq a27.6+14.12+511.21+464.30+511.39; minus -
      qq 0.6+23.12+511.21+471.30+511.39; plus -
      qq 0.6+35.12+143.21+511.30+511.39; point -
      qq a26.6+53.12+287.21+127.30+511.39; ten -
a15: it (s29) ,bs 9 ; [ten] if oldstate<3
      pm c34 ,gm 1c37 ; then number :=1
a16: it p ,qq (c84) ; [minus] sign:=sign+state (1,6,8)
a17: hv ra14 ,gr c54 ; store digit
      arm c37 ,nc ; if too many digits then begin
      qq (c53) t1 ; exp1:=exp1+1; goto a18 end;
      nc ,hv ra18 ;
      pm 1c37 ,tl 1 ; else
      gr c37 ,gm 1c37 ; number:=2×number
      tl 2 X ; M:=head(8×number)
      ar c54 ,gm c54 ; R:=tail(8×number)+digit
      ar 1c37 ,tl -39 ; M:=tail(10×number)+digit
      ar c37 ,ar c54 ; R:=mente+head(10×number)+digit
      gr c37 ,gm 1c37 ; number:=10×number+digit
a18: bs p-3 ,it -1 ; if state>3 then
      qq (c53) ,hh ra13 ; exp1:=exp1-1, goto next
a19: arm c49 ,bs p506 ; [digit]
      tk -30 ,hh ra17 ; not in exponent
      pm c54 ,ml s39 ; in exponent
      gm c54 ,hv ra14 ; exp2:=10×exp2+digit; goto next
a20: can p ,hv ra14 ; [termination] goto next
      can p-8 ,hh ra21 ; error exit
      vk 2c60 ,lk c67 ; call double number read 2
a21: can p-15 ,hsn s33 ; goto error track via basic read
      vk ,hs a40 ; goto read 2
a22: pp ,nt 2 ; restore p
      bt (c47) ,hv ra12 ; go back if more numbers
a23: qqf ,qq ; exit instruction

```

d a24=a22-a12, a25=a23-a12,

[the orders it (s29) in a15, ml s39 in 2a19 and hsn s33 in a21 refer to addresses in basic read.

Return after hsn s33 (from error track) by s-34]

[double number read, second track]

```

a30: mln (204) XV MTCT-205 ; head of 0.1, pos. 0
      cm (r102) VD IQB t409 ; tail of 0.1
a31: qq 160 ; head 10
      qq ; tail 10
a32: it s1 ,pt ra39 ; set return address
      can p-7 ,hh ra38 ; input ditto
      pi (c84) ,arn c54 ; indicator=sign, R:=exp2
      sm c54 LQA ; if neg. exponent then R:=-exp2
      ar c53 ,ga c54 ; exp2:=exp2+exp1
      ps ra30 V LT ; if neg. number then s:= addr.0.1
      ps ra31 ,it 3 ; else s:= addr. 10
      pa c53 t-4 ; exp1:=exp(konstant)
      ann c54 ,ga c54 ; exp2:=abs(exp2)
      arn c37 ,pm 1c37 ;
      nl ra34 ,tl -1 ; number pos.39 changed to number
      gr c37 ,it 78 ; pos. 0 + exponent
a33h: qq (ra34) ,gm 1c37 ; exponent:=exponent+78
      bt (c54) t-1 ; again:
      pm s1 ,hh r1 ; multiply exp2 times
      hv ra36 ,mkn c37 ; R:=tail(s)*head(number)
      pm s ,mk 1c37 ; R:=R+head(s)+tail(number)
      ml c37 ,nl ra35 ;
      tl -1 ,gr c37 ;
      arn c53 ;
a34: ar Q D ;
a35: ar Q D NO t3 ; exponent:=exponent+incr.+exp1
      ga ra34 V NO ; store exponent
      hv 0 NT ; goto spill
      hh ra33 NO ; if neg.overflow then goto again
      grn c37 ,grn 1c37 ; else number:=0
a36: pm c37 ,arn 1c37 ;
      hh ra37 XNRB ; RM:=number
      sm 1c37 ,tl -39 ; if neg. number
a37h: sr c37 ,nl r1 ; then number:=-number
      it ,qq (ra34) ; exponent:=exponent-1 or 2
      tl -10 ,gr (c36) ; store head(number)
      is (c36) ,gm s1 ; store tail(number)
      arn ra34 NZ ; if number ≠ 0 then
a38h: ga (c36) ,ud c89 ; store exponent
      vk c64 ,lk c67 ; restore running system
a39: [vk c64 ,hv Q ; order in running system, return]

```

d a40=c67+a32-a30

d c60=c60+e68+e68+e68

e

[TA 22.12.65]

b k=c60+e70, i=10, a27

[doublewrite, doubleoutput]

```
a:   pm   c90 XV           ; vy write
a1:  pm   ra5 X           ; pm 1023
    ps   (c46) ,pm s1     ; s:=last used; M:=layout description
a2:  gr   s1 X MRA        ; stack medium
    pm   (c47) DX t3      ; appetite+3
    ga   s1 ,hs c22       ; stack parameter counter
    arm  (c36) ,ps (c46)  ; layout value
a3:  hv   ra4 ,pm c68     ; next parameter
    ps   (c46) t1        ; last used:=last used+1
    arm  s ,ud s         ; restore output sum
    gm   1023 ,ud c89     ; restore medium
    pm   s-1 IRC         ; move return information
    gm   s MRC           ;
    pm   c43 ,gm c37     ; UV:=nonsense
    pm   s1 ,ud ra2      ; R:=layout, store medium and counter
a4:  pm   s2 ,it 1       ; M:=next parameter
    bt   (s1) ,hv c51    ; exit
    gr   s2             ; store layout
    gm   1c37 ,hs c22    ; UV1:=0
    ps   (c46) ,arm (c36) ; parameter value
a5:  sr   c43 ,pm 1023   ; -nonsense
    gm   c68 ,ud s1      ; if value=nonsense
    hh   ra3 LZ         ; then goto next parameter
    it   p ,pt c83      ; save p
    arnf s2 ,tkf -9     ; RA:=layout
    hs   c65 ,qq 1c60.29+1 ; call next track
    gt   sa26 ,sy 58     ; store layout, lower case
    arnf(c36) X ITB     ;
    ga   ra7 ,gm c37     ; store exponent
    is   (c36) ,arm s1   ; RM:=number
    hh   ra6 X NTB      ;
    is   (c36) ,srn s1   ; if number<0 then
a6:  sy   32 ,pa sa25    ; RM:=-number, minus
    sy   V NTB         ; else space
    tl   -39 ,sr c37    ;
a7:  pp   ,nl sa24      ; p:=exp.number
    pa   sa25 LZ t5     ; set a25
a8:  pi   32 ,hs s       ; set indicator, goto next track
a9:  qq   10 ,qq         ; konstant10, a-marked
    qq                     ; empty
```

d a1=a1-a, a10=a3-a8, a11=a9-a8, a27=1a6-a8

b k=e90, i=0

d i=c89

qq 193.9+c60.19

tdoublewrite;

qq 193.9+c60.19+a1.29

tdoubleoutput;

d e71=e71+e68+e68, e72=e72+e68+e68

d e89=e89+e68+c68+e68+e68+e68+e68

e

[doublewrite, doubleoutput, second track]

```

a12: gr c37 ,gm 1c37 ; store number
a13: pp p0 ,bs p-501 ; if p<-10 then
    pp p11 ,hh ra14 ; p:=p+11, goto a14
    qq (ra16) t-3 ; else begin exponent:=exponent-3
    mkn ra22 ,pm c37 ; number:=number*konstant
    mk ra23 ,ml ra22 ;
    nl ra13 ,pp p-9 ;
a14: hv ra12 ,pa ra20 ; goto a12 end;
a15: bs p ,hv ra17 ; if p=1 then goto print else
a16: qq t1 ; exponent:=exponent+1
    gr c37 ,mkn sa11 ; number:=number*10
    pm c37 ,ml sa11 ;
    nl r1 ,pp p9 ;
    pp p0 ,hv ra15 ; goto a15
a17: tl p-9 ,ga ra21 ; store first digit
    ca ,it 16 ; test for zero
a18: sy (ra21) ,pp 0 ; digit; p:=number of digits
    sy 59 ,hv ra21 ; point
a19: arn sa27 ,ud c83 ; reset p
    bt t512 ;
    hh sa10 ; return after exp. printing
    pp 3 ,nc (ra16) ; p:=3; if exp=0
    sy 60 ,sy 27 ; then
    ca (ra16) ,sy 0 ; else space
    an (ra16) DX IPA ; PA=0, R:=exp
    sy 58 ,mkn ra22 ; R:=exp*konstant
    bs (ra16) ,it 32 ; if exp<0 then minus
a20: sy ,pp p-1 ; else space, p:=p-1
    gr c37 ,mkn sa11 ;
    pm c37 ,ml sa11 ; number:=number*10
    ga ra21 ,nc ; store digit
    sy (ra21) V IPA ; if digit=0 then digit
    sy 16 V LPA ; else 0
    sy V NPA ; or space (before exponent)
a21: sr 0 D ; number:=number-digit
    bs p511 ,hv ra19 ; if p=1 then goto a19
    tl 9 ,hh ra20 ; else next digit
a22: gi r262 DXV NKA t147 ; head of constant(2 9/10 3)
a23: ly (r335) V RC t892 ; tail
    qq ; empty

```

d a24=a13-a12, a25=a16-a12, a26=a18-a12

d c60=c60+e68+e68

e  
s